INDEX

Name : Chandrapp Roshan Std. : Sec. :

Roll No. : Sub :

S.No.	Date	Title	~~Page~~ No. Marks	Teacher's Sign / Remarks
1.	9/8/24	8 - Queens Problem	9	
2.	16/8/24	DFS	9	
3.	23/8/24	DFS - water jug problem	10	
4.	30/8/24	A* Algorithm search	10	
5.	6/9/24	Implementation of Decision tree classification technique	10	
6.	27/9/24	Implementation of ANN for using python regression	10	
7.	4/10/24	Implementation of clustering algorithm	10	
8.	18/10/24	Minimax Algorithm	10	
9.	25/10/24	Introduction to prolog	10	
10.	8/11/24	Prolog Family tree	10	

completed

Ex. 1 8-Queens Problem Date :- 9/8

Aim:-

-1 to write a python program for 8-queens problem in jupiter Notebook

Program:-

```
def solve_8_queens(n):
    def is_safe(board, row, col):
        for i in range(row):
            if board[i] == col or
            board[i] - i == col - row or
            board[i] + i == col + row:
                return False
    return True

def place_queens(n, board, row):
    if row == n:
        return [board[:]]
    solutions = []
    for col in range(n):
        if is_safe(board, row, col):
            board[row] = col
            solutions.extend(place_queens(n,
            board, row + 1))
    return solutions

board = [-1]*n
return place_queens(n, board, 0)

n=8
solutions = solve_8_queens(n)
for solution in solutions:
```

print (solutions)

Output:

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

Result:

Thus the program for 8-queens for python code was run successfully.

Ex. 2 DFS Date: 16/8

Aim:

to write a python code for DFS using the jupiter notebook.

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(2,0)
```

```
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 program for depth first traversal was successfully executed.

Ex. 3 DFS / water jug problem Date: 30/3/24

Aim:

To write a python code for the water jug problem using the jupiter notebook.

Program:

```
def dfs_water_jug(capacity_jug1, capacity_jug2, derived_quantity):
    stack = [(0, 0)]
    visited = set()

    while stack:
        state = stack.pop()
        jug1, jug2 = state

        if (jug1, jug2) in visited:
            continue

        visited.add((jug1, jug2))

        if jug1 == derived_quantity or jug2 == derived_quantity:
            return [("Fill jug[i+1]" if r==0 else "Pour From jug[i+1] to jug[3-i-1]") for i in range(2)]

        next_states = []
        if jug1 < capacity_jug1:
            next_states.append((capacity_jug1, jug2))
        if jug2 < capacity_jug2:
            next_states.append((jug1, capacity_jug2))
        if jug1 > 0:
            next_states.append((0, jug2))
        if jug2 > 0:
            next_states.append((jug1, 0))
```

```
from next_state in next_states:
        stack.append(next_state)

return "No solution Found"
```

Output:

Solution : (4 1 2)

Result:-

Thus the program for water jug problem using python was successfully executed.

Ex. 4 A* algorithm Date: 30/8

Aim:

to write a python code for A* star Algorithm using jupiter Notebook.

Program:

```
import heapq
def heuristic (a,b):
    return (abs (a[0] - b[0]) + abs (a[1] - b[1])
def a_star_search (grid, start, goal):
    directions = [(0,1), (1,0), (0,-1), (-1,0)]
open_set = []
heapq.heappush (open_set, (0, start))
g_score = {start: 0}
f_score = {start : heuristic (start, goal)}
came_from = {}
while open_set :
    _, current = heapq.heappop (open_set)
if current == goal:
    path = []
    while current in came_from:
        path.append (current)
        current = came_from [current]
    path.append (start)
    path.reverse()
    return path
```

```
for direction in directions:
    neighbour = (current[0] + direction[0], current[1]
    + direction[1])

if 0<= neighbour[0] < len(grid) and 0<=neighbour
[1] <len(grid[0]) and grid[neighbour[0][neigh
[1]]==0:
    tentative_g_score = g_score[current]+1

    if tentative_g_score < g_score.get(neighbour, float
('inf')):
    came_from[neighbour] = current
    g_score[neighbour] = tentative_g_score
    f_score[neighbour] = tentative_g_score +
    heuristic(neighbor, goal)
    if neighbour not in [item[1] for item
    in open_set]:
        heapq.heappush(open_set, (f_score[neighbor
    neighbour))
    return None
grid = [
    [0,1,0,0,0]
    [0,1,0,1,0]
    [0,0,0,1,0]
    [0,1,1,1,0]
    [0,0,0,0,0]
]
```

```
start = (0,0)
goal = (4,4)
path = a_star_search (grid, start, goal)
if path:
    print ("path found:", path)
else:
    print ("No path found.")
```

Output:

The path is

→ (8,0) → (7,0) → (6,0) → (5,0) → (4,1)
→ (3,2) → (2,1) → (1,0) → (0,0)

Result:-

Thus the program for A* star Algorithm was successfully executed.

Ex-5 Date:- 15/9

Aim:

→ to write a code for implementation of decision tree classification technique using jupiter notebook.

Program:-

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
df = pd.read_csv('your_dataset.csv')
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
```

print (confusion_matrix (y_test, y_pred))

Output:

DATA INFO

Dataset length : 625

Dataset shape : (625, 5)

Dataset :

	0	1	2	3	4
0	B	1	1	1	1
1	R	1	1	1	2
2	R	1	1	1	3
3	R	1	1	1	4
4	R	1	1	1	5

Result:

Thus the program was successfully executed.

Ex-6 Best set of Parameters Date:- 29/

Aim:-

to write the program for ANN using python registered on jupiter Notebook.

Program:-

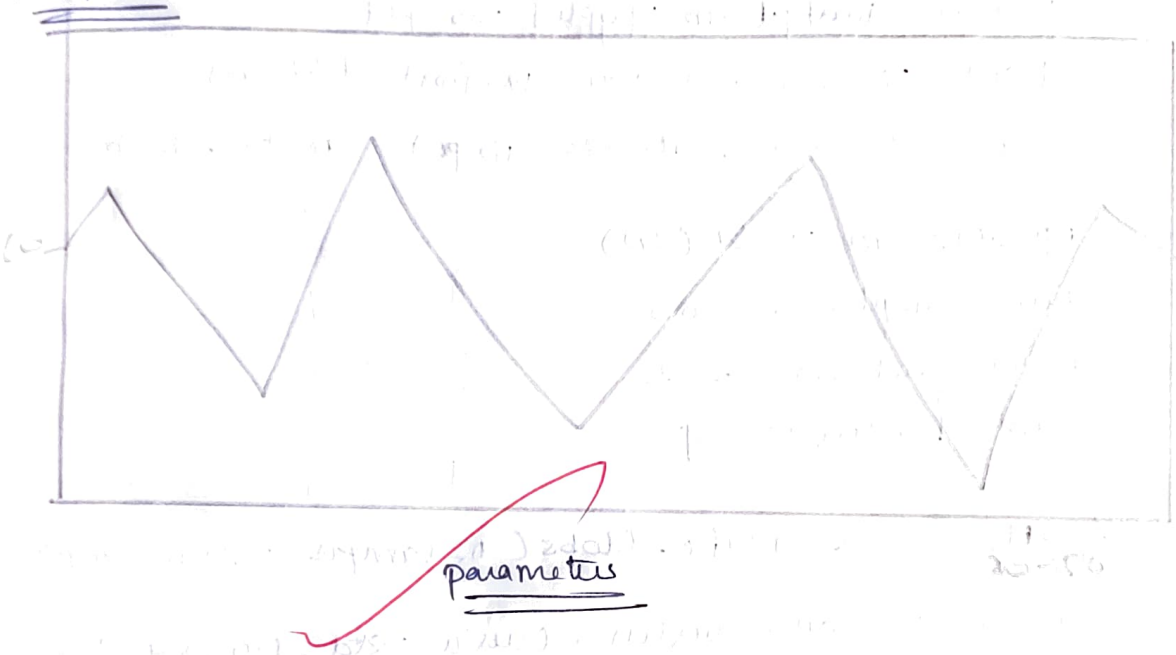
```
def FunctionFindBestParams(X_train, y_train, X_test,
y_test):

batch_size_list = [5, 10, 15, 20]
batch epoch_list = [5, 10, 50, 100]

import pandas as pd
SearchResultsData = pd.DataFrame(['TrailNumber',
'Parameters', 'Accuracy'])

TrailNumber = 0
for batch_size_trail in batch_size_list:
    for epochs_trail in epoch_list:
        TrailNumber += 1
    model = Sequential()
    model.add(Dense(units=5, input_dim=X_tr
    Shape[1], kernel_initializer='normal',
    activation = 'relu'))

    model.add(Dense(1, kernel_initializer='norma
    mape = np.mean(100 * (np.abs(y_test-model
    .predict(X_test))/y_test))

print(TrialNumber, 'Parameters:', 'batch_size'
batch_size_trial, 'epochs', epochs_tri
```

'Accuracy' : (100-MAPE)

return (Search Results Data)

ResultsData = Function find Best params (x_train, y_train, x_test, y_test)

Output:-



Result:-

The program has been successfully executed.

Ex. 7 K-means Clustering Page :- 1/10

Aim :-

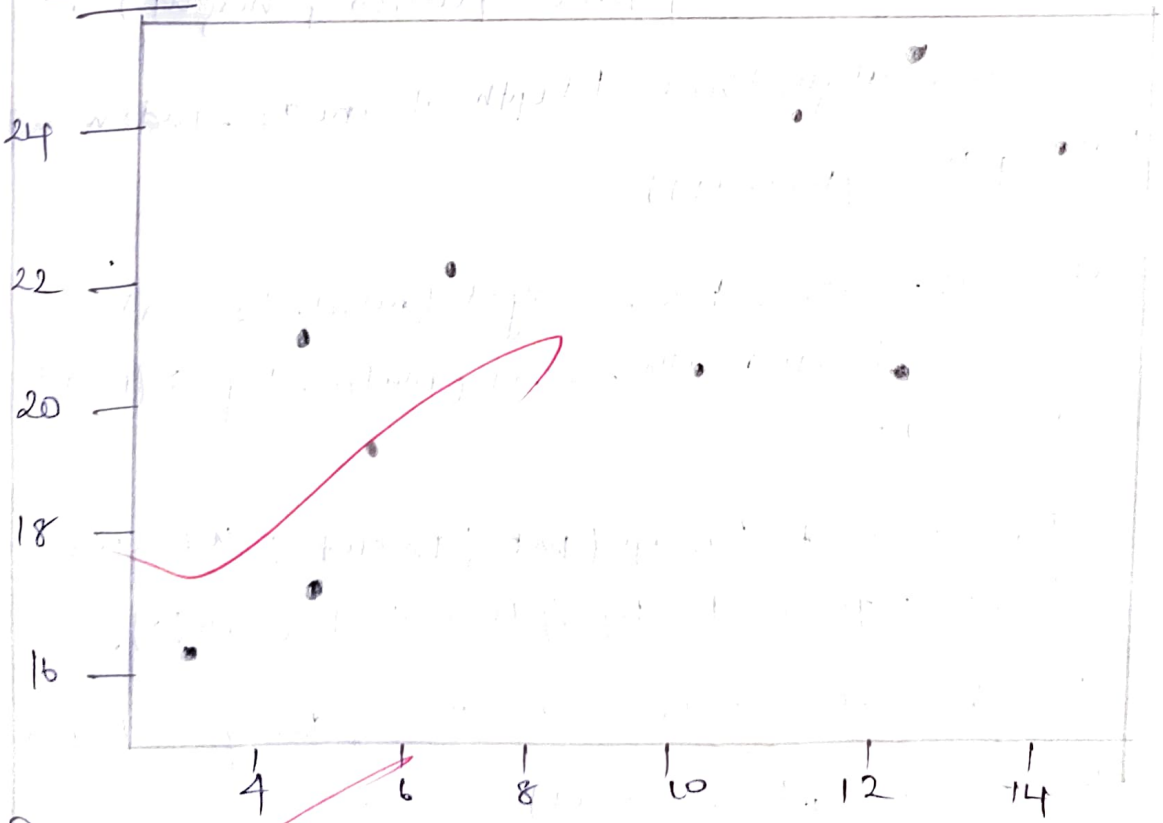
to write a python for k-means clustering algorithm with jupiter Notebook.

Program :-

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

np.random.seed(42)
num_samples = 300
num_features = 2
num_clusters = 4

X, y_true = make_blobs(n_samples = num_samples
centers = num_clusters, cluster_std = 1.0, random
state = 42)

kmeans = KMeans(n_clusters = num_clusters, random_
state = 42)
kmeans.fit(X)

y_kmeans = kmeans.predict(X)
plt.figure(figsize = (8,6))
plt.scatter(X[:,0], X[:,1], c = y_kmeans, s
cmap = 'viridis')
```

```
centers = kmeans.cluster_centers_
plt.scatter(centers[:,0], centers[:,1],
c='red', s=200, alpha=0.75, label='Centers')
plt.title("K-means Clustering")
plt.xlabel("Feature 1")
plt.xlabel("Feature 2")
plt.legend()
plt.show()

print("Cluster centers:\n", centers)
print("Inertia (sum of squared distances of
samples to their closet cluster center):",
kmeans.inertia_)
```

Output:-



Result:-

Thus the program was successfully executed..

Ex. 8 Minimax Algorithm Date:- 18/4

Aim:-

to write a program for minimax algorithm for in jupiter Notebook.

Program:-

```
import Math
def minimax (depth, node_index,
is_maximizer, scores, height)
    if depth == height
        return scores [node_index]
    if is_maxlayer:
        return max (minimax, (depth+1,
node_index * 2, False, scores, height),
minimax algorithm (depth + 1, node_index *
False, score (height))

def calculate_tree_height (num_leaves):
    return math.ceil (math.log 2 (num_
leaves))

scores = list (map (int (input ("Enter the
score seperated by spaces :" ).split ()).
tree.height = calculate_tree_height (0,0)
True, scores, tree_height)
Optimal_score = minimax (0,0, True, tree
tree_height)
```

print ("The optimal score is : {optimal score}")

Output:

Enter the scores seperated by spaces

-1 4 2 6 -3 -5 0 7

the optimal score is 4

Result :-

Thus the program was succesfully executed.

Ex. 9 Introduction to Prolog Date:- 25/

Aim:-

To learn Prolog terminologies and write basic programs

Program:-

1. Atomic Terms:-

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

2. Variable.

Variables are strings ~~are~~ of letters, digits and the underscore, starting with a capital letter or an underscore

Ex.
Dog
Apple-420

3. Compound terms

Compound terms are made up of a prolog atom and a number of arguments enclosed in parenthesis and seperated by commas

4. Facts

A fact is a predicate followed by a dot.

5. Rules

A rule consists of a head and a body. A body is predicates separated by commas

Source Code

KB1:-

woman (mia)
woman (jody)
woman (yolanda)
plays AirGuitar (jody).
party

Output:-

? - woman(mia)
true

?- plays AirGuitar (mia).
False

?- party
True

KB2 :-

happy (yolanda).

listen 2music (mia).

listens2music (yolanda) :- happy (yolanda).

plays AirGuitar (mia) :- listen 2music (mia).

plays AirGuitar (yolanda) :- listens 2music (yolanda).

Output :-

?- playsAirGuitar (mia)

true

?- playsAirGuitar (yolanda)

true

Result :-

Thus the program for prolog was successfully executed.

Ex.10 PROLOG Family tree Date: 8/11/24

Aim:

to write a program for prolog family tree

Source code

```
/* Facts :: */

male (peter).
male (john).
male (chris).
male (kevin).

female (betty).
female (jeny)
female (lucia)
female (helen).

parent of (chris, peter)
parent of (chris, betty)
parent of (helen, peter).
parent of (helen, betty)
parent of (kevin, chris)
parent of (kevin, betty).
parent of (jeny, john)
parent of (jeny, helen).

/* Rules :: */

/* Son. parent
   Son. grandparent */

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

Mother (X, Y) :- female (X), parent of (X, Y).
```

grandfather (x,y) :- male (y), parent of (x,z), parent of (z,y)

grandmother (x,y) :- female (x), parent of (x,z), parent of (z,y)

brother (x,y) :- male (y), father (x,z), father (y,w), z == w

sister (x,y) :- female (y), father (x,z), father (y,w), z == w

Output:-

Result:-

Thus the program was successfully created.