**Name: Gaurang Vaghela**

**Rollno: TEAD-22561**

**ANN (SL-II)**

**Practical 8  
Problem Statement:** Implement a python program to illustrate ART neural network.

**Code:**

import numpy as np

class ART1:

def \_\_init\_\_(self, input\_size, num\_categories, vigilance):

# Initialize the parameters: input size, number of categories, vigilance parameter

self.input\_size = input\_size

self.num\_categories = num\_categories

self.vigilance = vigilance

# Initialize weight matrix with ones

self.weights = np.ones((num\_categories, input\_size))

def complement\_code(self, input\_vector):

# Generate the complement-coded input by concatenating the input with its complement

return np.concatenate([input\_vector, 1 - input\_vector])

def match\_category(self, input\_vector):

# Compute matching scores for each category

scores = np.dot(self.weights, input\_vector) # Dot product of weights and input

norms = np.sum(input\_vector) # Sum of input vector (norm of input)

return scores / norms # Normalized matching score

def train(self, inputs):

# Train the network with given input vectors

for input\_vector in inputs:

input\_vector = self.complement\_code(input\_vector) # Apply complement coding

while True:

match\_scores = self.match\_category(input\_vector) # Calculate match scores

chosen\_category = np.argmax(match\_scores) # Choose category with max score

# Vigilance test to check if the category matches sufficiently

if np.sum(np.minimum(input\_vector, self.weights[chosen\_category])) / np.sum(input\_vector) >= self.vigilance:

# Update weights based on the minimum of the input vector and the selected category's weights

self.weights[chosen\_category] = np.minimum(input\_vector, self.weights[chosen\_category])

break # Exit the loop after successful learning

else:

# Reset the category by setting its weight to zero if vigilance test fails

self.weights[chosen\_category] = np.zeros\_like(self.weights[chosen\_category])

def predict(self, input\_vector):

# Make a prediction for a given input vector

input\_vector = self.complement\_code(input\_vector) # Apply complement coding

match\_scores = self.match\_category(input\_vector) # Calculate match scores

return np.argmax(match\_scores) # Return the category with the highest match score

# Example usage

inputs = np.array([[1, 0, 0], [0, 1, 0], [1, 1, 0], [0, 0, 1]]) # Sample input vectors

art = ART1(input\_size=6, num\_categories=4, vigilance=0.7) # Initialize ART1 network with vigilance of 0.7

art.train(inputs) # Train the network on the input data

# Print the predictions for each input

print("Predictions:")

for input\_vector in inputs:

print(f"Input: {input\_vector}, Predicted Category: {art.predict(input\_vector)}")

**Output:**

