**CUCKOO SEARCH ALGORITHM**

import numpy as np

# Simple neural network: 1 input, 1 output (linear)

def nn\_output(x, w):

return x \* w # linear weight

# Fitness function: Mean Squared Error

def fitness(w, X, Y):

y\_pred = nn\_output(X, w)

return np.mean((Y - y\_pred)\*\*2)

# Levy flight step

def levy(beta=1.5):

u = np.random.randn()

v = np.random.randn()

sigma = (np.math.gamma(1+beta) \* np.sin(np.pi\*beta/2) /

(np.math.gamma((1+beta)/2) \* beta \* 2\*\*((beta-1)/2)))\*\*(1/beta)

return u \* sigma / abs(v)\*\*(1/beta)

# Cuckoo Search Algorithm with iteration output

def cuckoo\_search\_nn(X, Y, n\_nests, max\_iter, Pa=0.25, alpha=0.1):

nests = np.random.uniform(-5,5,n\_nests)

fitnesses = np.array([fitness(w, X, Y) for w in nests])

best\_idx = np.argmin(fitnesses)

best\_w = nests[best\_idx]

best\_f = fitnesses[best\_idx]

for t in range(1, max\_iter+1):

for i in range(n\_nests):

w\_new = nests[i] + alpha \* levy()

f\_new = fitness(w\_new, X, Y)

if f\_new < fitnesses[i]:

nests[i] = w\_new

fitnesses[i] = f\_new

for i in range(n\_nests):

if np.random.rand() < Pa:

nests[i] = np.random.uniform(-5,5)

fitnesses[i] = fitness(nests[i], X, Y)

idx = np.argmin(fitnesses)

if fitnesses[idx] < best\_f:

best\_f = fitnesses[idx]

best\_w = nests[idx]

# Print best solution for this iteration

print(f"Iteration {t}: Best weight = {best\_w:.4f}, Minimum error = {best\_f:.6f}")

return best\_w, best\_f

# ---------------- User Input ----------------

X = input("Enter input values X (comma-separated): ")

Y = input("Enter target values Y (comma-separated): ")

n\_nests = int(input("Enter number of nests: "))

max\_iter = int(input("Enter number of iterations: "))

X = np.array([float(x.strip()) for x in X.split(",")])

Y = np.array([float(y.strip()) for y in Y.split(",")])

# Run Cuckoo Search

best\_w, best\_f = cuckoo\_search\_nn(X, Y, n\_nests=n\_nests, max\_iter=max\_iter)

print("\nFinal Best weight:", best\_w)

print("Final Minimum error:", best\_f)

