Nidhi D B – 1BM23CS211

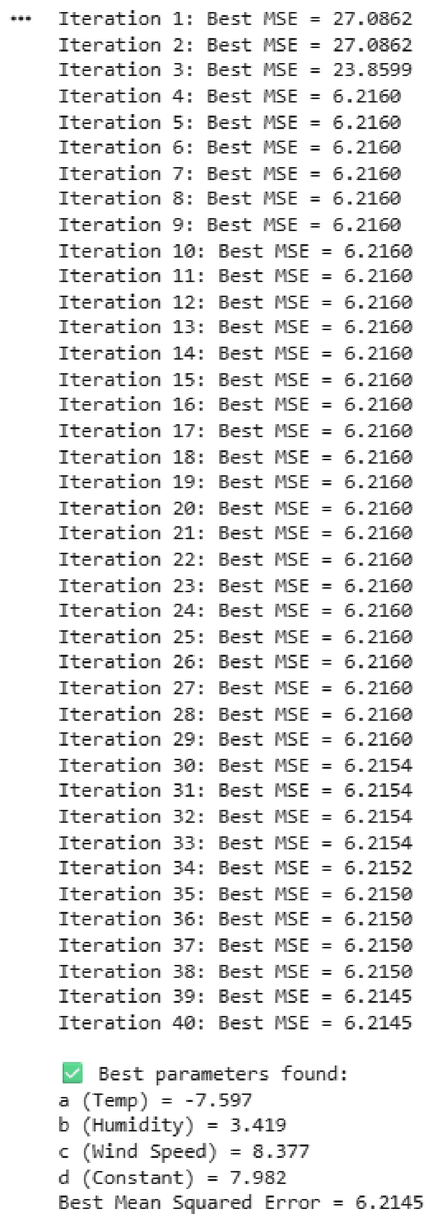
QUESTION: **Model and predict environmental systems (e.g. pollution levels, climate change) by optimizing parameters and equations based on historical data and simulation results**
CODE:

```
import numpy as np
import matplotlib.pyplot as plt

# ----------------------------
# Step 1: Create simple fake data
# ----------------------------
np.random.seed(42)  # for repeatable results

# Example environmental data
temperature = np.linspace(10, 35, 30)        # 30 temperature readings
humidity = np.linspace(30, 80, 30)           # 30 humidity readings
wind_speed = np.linspace(1, 10, 30)          # 30 wind speed readings

# True (hidden) relationship in nature
# Pollution = 1.5*Temp + 0.8*Humidity - 2.0*WindSpeed + 5 + noise
pollution_actual = (
    1.5 * temperature +
    0.8 * humidity -
    2.0 * wind_speed +
    5 +
    np.random.randn(30) * 3  # small random noise
)

# ----------------------------
# Step 2: Define our pollution model
# ----------------------------
def pollution_model(a, b, c, d, temp, hum, wind):
    return a * temp + b * hum + c * wind + d

# ----------------------------
# Step 3: Define how "good" our model is (Mean Squared Error)
# ----------------------------
def fitness(a, b, c, d):
    predicted = pollution_model(a, b, c, d, temperature, humidity,
wind_speed)
    mse = np.mean((pollution_actual - predicted) ** 2)
    return mse  # smaller is better

# ----------------------------
# Step 4: Initialize PSO parameters
# ----------------------------
num_particles = 15       # number of guesses
```

```python
max_iter = 40              # number of loops

# Randomly guess initial values for a, b, c, d
positions = np.random.uniform(-5, 5, (num_particles, 4))  # [a, b, c,
d]
velocities = np.zeros((num_particles, 4))

# Each particle remembers its best position
personal_best = positions.copy()
personal_best_value = np.array([fitness(*p) for p in positions])

# Global best (best among all)
best_idx = np.argmin(personal_best_value)
global_best = personal_best[best_idx]
global_best_value = personal_best_value[best_idx]

# ----------------------------
# Step 5: PSO main loop
# ----------------------------
for it in range(max_iter):
    for i in range(num_particles):
        # Random learning factors
        r1, r2 = np.random.rand(4), np.random.rand(4)

        # Update velocity (how the particle moves)
        velocities[i] = (0.5 * velocities[i] +
                         1.5 * r1 * (personal_best[i] - positions[i]) +
                         1.5 * r2 * (global_best - positions[i]))
        # Update position (new guess)
        positions[i] += velocities[i]

        # Calculate new fitness
        current_value = fitness(*positions[i])
        if current_value < personal_best_value[i]:
            personal_best_value[i] = current_value
            personal_best[i] = positions[i]

    # Update global best
    best_idx = np.argmin(personal_best_value)
    if personal_best_value[best_idx] < global_best_value:
        global_best_value = personal_best_value[best_idx]
        global_best = personal_best[best_idx]

    print(f"Iteration {it+1}: Best MSE = {global_best_value:.4f}")

# ----------------------------
# Step 6: Show results
# ----------------------------
```

```python
print("\n✅ Best parameters found:")
print(f"a (Temp) = {global_best[0]:.3f}")
print(f"b (Humidity) = {global_best[1]:.3f}")
print(f"c (Wind Speed) = {global_best[2]:.3f}")
print(f"d (Constant) = {global_best[3]:.3f}")
print(f"Best Mean Squared Error = {global_best_value:.4f}")

# Predict pollution with best parameters
predicted = pollution_model(global_best[0], global_best[1],
                            global_best[2], global_best[3],
                            temperature, humidity, wind_speed)
```

OUTPUT:



```
Iteration 1: Best MSE = 27.0862
Iteration 2: Best MSE = 27.0862
Iteration 3: Best MSE = 23.8599
Iteration 4: Best MSE = 6.2160
Iteration 5: Best MSE = 6.2160
Iteration 6: Best MSE = 6.2160
Iteration 7: Best MSE = 6.2160
Iteration 8: Best MSE = 6.2160
Iteration 9: Best MSE = 6.2160
Iteration 10: Best MSE = 6.2160
Iteration 11: Best MSE = 6.2160
Iteration 12: Best MSE = 6.2160
Iteration 13: Best MSE = 6.2160
Iteration 14: Best MSE = 6.2160
Iteration 15: Best MSE = 6.2160
Iteration 16: Best MSE = 6.2160
Iteration 17: Best MSE = 6.2160
Iteration 18: Best MSE = 6.2160
Iteration 19: Best MSE = 6.2160
Iteration 20: Best MSE = 6.2160
Iteration 21: Best MSE = 6.2160
Iteration 22: Best MSE = 6.2160
Iteration 23: Best MSE = 6.2160
Iteration 24: Best MSE = 6.2160
Iteration 25: Best MSE = 6.2160
Iteration 26: Best MSE = 6.2160
Iteration 27: Best MSE = 6.2160
Iteration 28: Best MSE = 6.2160
Iteration 29: Best MSE = 6.2160
Iteration 30: Best MSE = 6.2154
Iteration 31: Best MSE = 6.2154
Iteration 32: Best MSE = 6.2154
Iteration 33: Best MSE = 6.2154
Iteration 34: Best MSE = 6.2152
Iteration 35: Best MSE = 6.2150
Iteration 36: Best MSE = 6.2150
Iteration 37: Best MSE = 6.2150
Iteration 38: Best MSE = 6.2150
Iteration 39: Best MSE = 6.2145
Iteration 40: Best MSE = 6.2145

✅ Best parameters found:
a (Temp) = -7.597
b (Humidity) = 3.419
c (Wind Speed) = 8.377
d (Constant) = 7.982
Best Mean Squared Error = 6.2145
```