**Artificial Intelligence**



**Bat Algorithm**



***Under Supervision of:***

**Dr.Osman Ali**

Dept. of Computer Science, Faculty of Science

*Team members:*

**Eman Adel**

**Eriny Hany**

**Fatma Ashraf**

**Fatma Khalid**

**Fauzia Medhat**

**Radwa Saleh**

**Noha Sayed**

**Table of Contents**

[**A Nature-Inspired Optimization Technique** 3](#_Toc185290911)

[**Introduction** 3](#_Toc185290912)

[**Inspiration** 3](#_Toc185290913)

[**Echolocation** 3](#_Toc185290914)

[**Key behaviors mimicked in BA** 3](#_Toc185290915)

[**Key Features of the Bat Algorithm** 3](#_Toc185290916)

[**1)** **Echolocation Behavior** 3](#_Toc185290917)

[**2)** **Movement of Bats** 4](#_Toc185290918)

[**3)** **Frequency** 4](#_Toc185290919)

[**4)** **Loudness** 4](#_Toc185290920)

[**The Bat Algorithm is based on two strategies** 5](#_Toc185290921)

[**Exploration** 5](#_Toc185290922)

[**Exploitation** 5](#_Toc185290923)

[**How the Bat Algorithm Works** 5](#_Toc185290924)

[**Initialization** 5](#_Toc185290925)

[**Movement and Update** 5](#_Toc185290926)

[**Pseudocode** 7](#_Toc185290927)

[**Applications of the Bat Algorithm** 8](#_Toc185290928)

[**Advantages of the Bat Algorithm** 8](#_Toc185290929)

[**Disadvantages of the Bat Algorithm** 9](#_Toc185290930)

[**Mathematical Formulation** 9](#_Toc185290931)

[**Dataset** 9](#_Toc185290932)

[ **Heart** 9](#_Toc185290933)

[ **Air quality** 18](#_Toc185290934)

[ **latest covid-19** 28](#_Toc185290935)

[ **customer churn** 38](#_Toc185290936)

[**Code steps followed** 48](#_Toc185290937)

[**Conclusion** 49](#_Toc185290938)

# **A Nature-Inspired Optimization Technique**

The Bat Algorithm (BA) is a metaheuristic optimization algorithm inspired by the echolocation behavior of bats.

We will explore its workings, key features, applications, and advantages and disadvantages.

# **Introduction**

The Bat Algorithm (BA) is inspired by the echolocation behavior of bats.

Bats use sound pulses to locate prey and avoid obstacles.

BA mimics this behavior to solve optimization problems.

# **Inspiration**

# **Echolocation**

Bats emit sound pulses and analyze echoes to determine the location, size, and distance of objects.

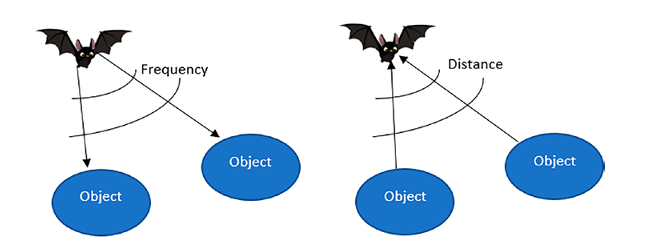
# **Key behaviors mimicked in BA**

* Adjusting sound frequency and loudness for exploration and exploitation.
* Moving towards the best-known solution while refining local searches.

# **Key Features of the Bat Algorithm**

## **Echolocation Behavior**

Bats emit sound pulses and use the echoes to estimate distance to objects.



**Figure 1. Echolocation in bats**

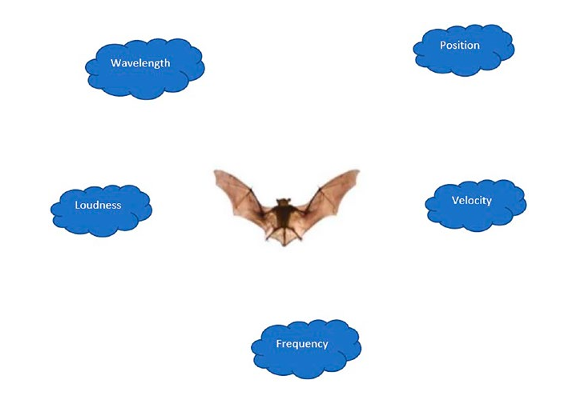
## **Movement of Bats**

Bats adjust positions based on distance from the best solution and random velocity variation.

## **Frequency**

Influences velocity and movement, higher frequency means smaller steps.

## **Loudness**

Decreases as bats get closer to their target, signifying solution refinement.

**Figure 2. Various dimensions for bats**

# 

# **The Bat Algorithm is based on two strategies**

## **Exploration**

* Searching broadly in the solution space.
* Controlled by high loudness and low pulse rates.

## **Exploitation**

* Focusing search near the current best solution.
* Loudness decreases and pulse rate increases near the target.

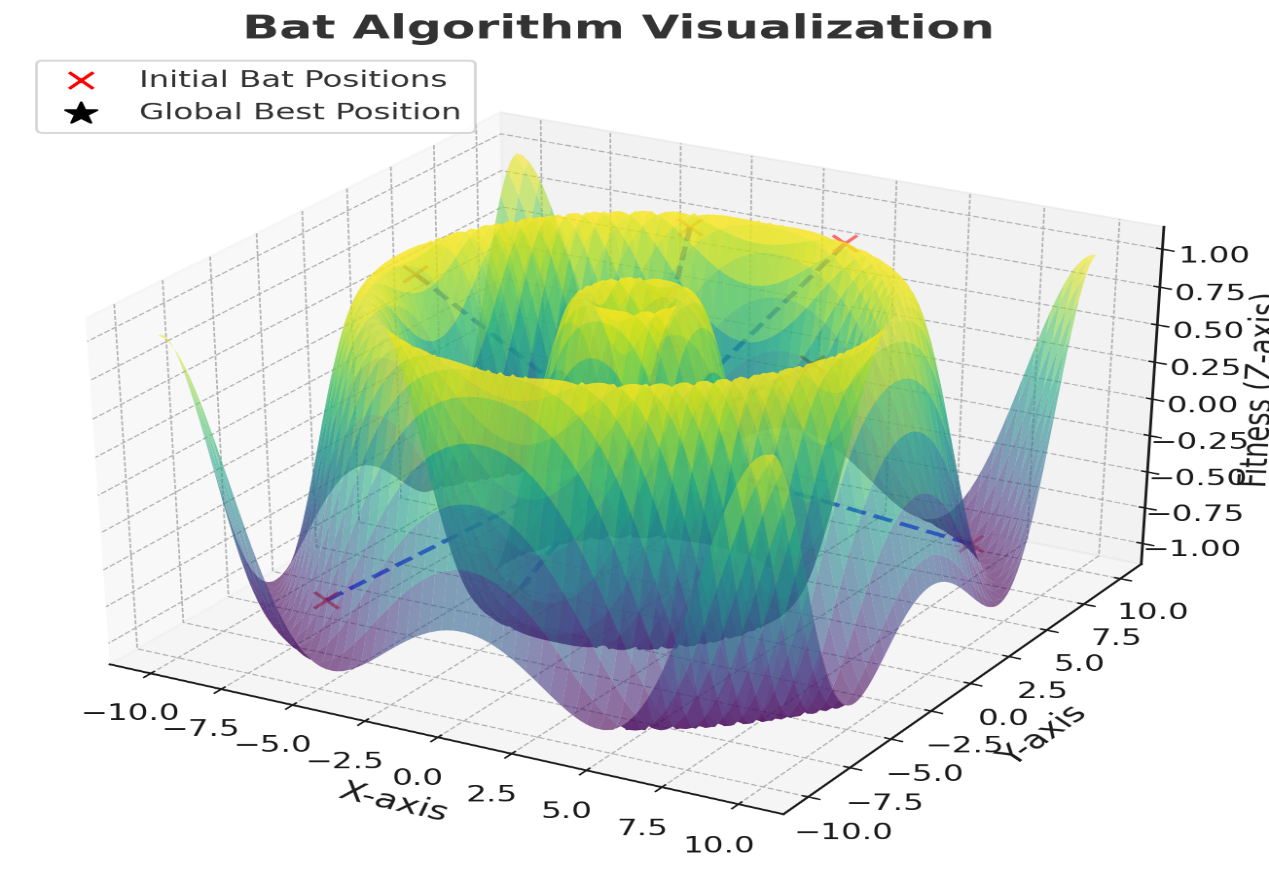
# **How the Bat Algorithm Works**

## **Initialization**

The algorithm begins by randomly initializing a population of bats within the search space. Each bat is assigned initial values for frequency, loudness, and pulse emission rate, crucial parameters for their movement and echolocation.

## **Movement and Update**

Bats move towards the best solution found so far, guided by their dynamically adjusted frequency. This frequency balance exploration of the search space and exploitation of promising regions.



**Figure 3. Bat Algorithm visualization**

**Key Points**

* Initialization:The bats' positions and velocities are set randomly within the search space.
* Frequency (f):Controls the step size of movements (global or local search).
* Loudness (A):Gradually decreases to refine solutions near the optimum.
* Pulse Rate (r):Increases to favor local exploitation near the best solution.
* Stopping Criterion:The algorithm terminates after a fixed number of iterations or when convergence is achieved.

# **Pseudocode**

1. Initialize the bat population (positions and velocities) randomly.

2. Assign initial values for frequency (f), loudness (A), and pulse rate (r).

3. Define the objective function (fitness) to optimize.

4. Evaluate the fitness of each bat and find the current best solution (X\*).

5. Repeat until the stopping criterion is met (e.g., max iterations):

a. For each bat i in the population:

i. Adjust the frequency (f) using:

f\_i = f\_min + (f\_max - f\_min) \* rand

ii. Update the velocity (v) and position (x) using:

v\_i^t = v\_i^{t-1} + (x\_i^t - X\*) \* f\_i

x\_i^t = x\_i^{t-1} + v\_i^t

iii. Perform a local search (with a probability proportional to r):

If rand > r\_i:

Generate a new solution near the best solution:

x\_new = X\* + ε \* A\_i (ε is a random number in [-1, 1])

iv. Evaluate the new solution:

If the new solution improves the fitness AND satisfies loudness criteria:

Accept the new solution.

Update A\_i and r\_i using:

A\_i^{t+1} = α \* A\_i^t

r\_i^{t+1} = r\_i^t \* (1 - exp(-γ \* t))

b. Update the global best solution (X\*) if needed.

6. Return the global best solution (X\*) as the optimal solution.

# **Applications of the Bat Algorithm**

* Engineering Design

Optimizing structural designs, mechanical systems, and electrical circuits.

* Control Systems

Designing controllers for complex systems.

* Image Processing

Image denoising, segmentation, and feature extraction.

* Machine Learning

Feature selection, classification, clustering problems, parameter tuning, and neural network training.

# **Advantages of the Bat Algorithm**

1. Robustness

Handles complex and noisy optimization problems.

1. Efficiency

Converges faster than other metaheuristic algorithms.

1. **Versatility**

Can be applied to a wide range of optimization problems.

# **Disadvantages of the Bat Algorithm**

* Local Minima

May get stuck in local minima, particularly in complex landscapes.

* Parameter Sensitivity

Performance may be sensitive to the choice of parameters.

# **Mathematical Formulation**

* Position Update

xit+1 = xit + vit

* Velocity Update

vit+1 = vit + α(xbestt - xit) + β ϵ

* Frequency Update

fit+1 = fmin + γ(fmax - fmin)

* Loudness Update

Ait+1 = Ait α

# **Dataset**

# **Heart**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

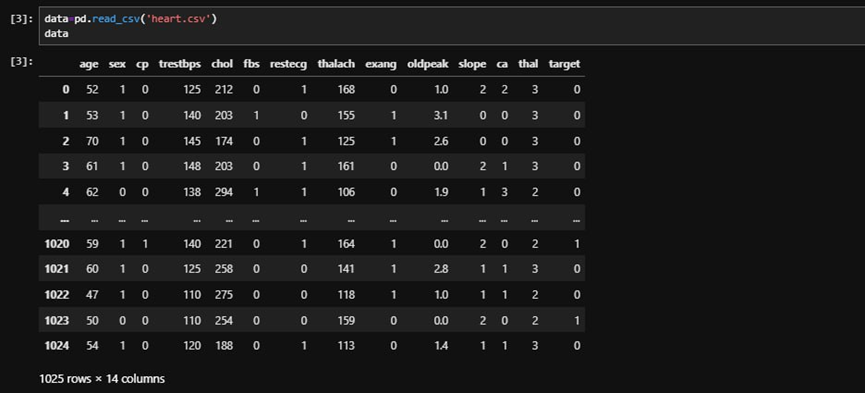
from sklearn.datasets import make\_classification

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

data=pd.read\_csv('heart.csv')

data

**T****he data after executing the code:**

**# Features and Target**

X = data.drop(columns=["target"])

y = data["target"]

**# Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**# Bat Algorithm Parameters**

num\_bats = 10

max\_iter =20

loudness = 0.6

pulse\_rate = 0.5

frequency\_min, frequency\_max = 0, 2

**# Fitness Function**

def fitness\_function\_with\_metrics(solution):

selected\_features = np.where(solution > 0.5)[0]

if len(selected\_features) == 0:

return 0, 0, 0, 0 # Return 0 for accuracy, precision, recall, and F1 if no features are selected

**# Train model on selected features**

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train.iloc[:, selected\_features], y\_train)

preds = clf.predict(X\_test.iloc[:, selected\_features])

**# Calculate metrics**

accuracy = accuracy\_score(y\_test, preds)

precision = precision\_score(y\_test, preds)

recall = recall\_score(y\_test, preds)

f1 = f1\_score(y\_test, preds)

return accuracy, precision, recall, f1

**# Initialize bats**

positions = np.random.rand(num\_bats, X.shape[1]) # Random positions in [0,1]

velocities = np.random.uniform(-1, 1, (num\_bats, X.shape[1])) # Random velocities

best\_global = positions[np.random.randint(0, num\_bats)] # Randomly select a bat

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = fitness\_function\_with\_metrics(best\_global)

**# Bat Algorithm**

for t in range(max\_iter):

for i in range(num\_bats):

**# Calculate frequency and update velocity and position**

frequency = frequency\_min + (frequency\_max - frequency\_min) \* np.random.rand()

velocities[i] += (positions[i] - best\_global) \* frequency

positions[i] = np.clip(positions[i] + velocities[i], 0, 1) # Ensure positions stay in range

**# Local search**

if np.random.rand() > pulse\_rate:

positions[i] = np.clip(best\_global + np.random.normal(0, 0.1, size=X.shape[1]), 0, 1)

**# Evaluate fitness**

accuracy, precision, recall, f1 = fitness\_function\_with\_metrics(positions[i])

if accuracy > best\_global\_accuracy and np.random.rand() < loudness:

best\_global = positions[i]

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = accuracy, precision, recall, f1

**# Update loudness and pulse rate**

loudness = max(0.1, loudness \* 0.95)

pulse\_rate = min(1.0, pulse\_rate \* 1.05)

**# Print progress**

print(f"Iteration {t + 1}:")

print(f" Best Accuracy = {best\_global\_accuracy:.4f}")

print(f" Best Precision = {best\_global\_precision:.4f}")

print(f" Best Recall = {best\_global\_recall:.4f}")

print(f" Best F1 Score = {best\_global\_f1:.4f}")

selected\_features = np.where(best\_global > 0.5)[0]

print(" Selected Features:", X.columns[selected\_features].tolist())

print()

**# Final Results**

print("Final Results:")

print(f"Best Accuracy: {best\_global\_accuracy:.4f}")

print(f"Best Precision: {best\_global\_precision:.4f}")

print(f"Best Recall: {best\_global\_recall:.4f}")

print(f"Best F1 Score: {best\_global\_f1:.4f}")

print("Selected Features:", X.columns[selected\_features].tolist())

**OUTPUT:**

Iteration 1:

Best Accuracy = 0.9903

Best Precision = 1.0000

Best Recall = 0.9799

Best F1 Score = 0.9898

Selected Features: ['age', 'sex', 'trestbps', 'chol', 'exang', 'slope', 'thal']

Iteration 2:

Best Accuracy = 0.9903

Best Precision = 1.0000

Best Recall = 0.9799

Best F1 Score = 0.9898

Selected Features: ['age', 'sex', 'trestbps', 'chol', 'thal']

Iteration 3:

Best Accuracy = 0.9903

Best Precision = 1.0000

Best Recall = 0.9799

Best F1 Score = 0.9898

Selected Features: ['age', 'sex', 'trestbps', 'chol']

Iteration 4:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['age', 'sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 5:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 6:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 7:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 8:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 9:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 10:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 11:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 12:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 13:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 14:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 15:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 16:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'chol', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 17:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 18:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 19:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'fbs', 'oldpeak', 'slope', 'ca']

Iteration 20:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['sex', 'cp', 'fbs', 'oldpeak', 'slope', 'ca']

**Final Results:**

Best Accuracy: 1.0000

Best Precision: 1.0000

Best Recall: 1.0000

Best F1 Score: 1.0000

Selected Features: ['sex', 'cp', 'fbs', 'oldpeak', 'slope', 'ca']

# **Air quality**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_classification

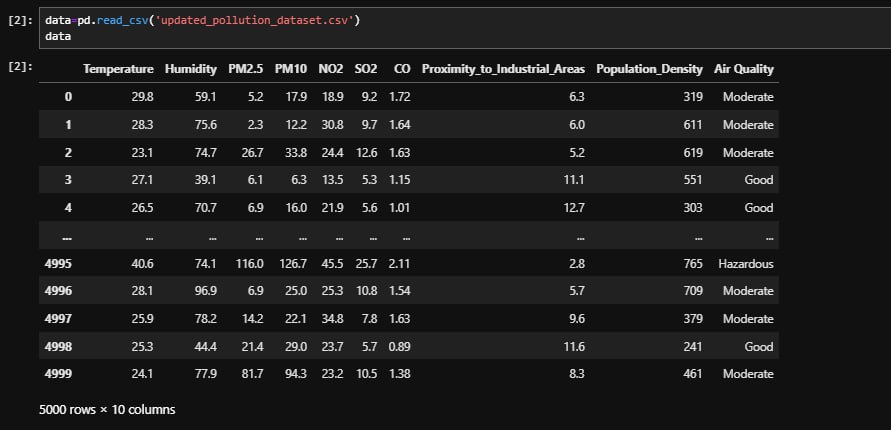
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

from sklearn.preprocessing import LabelEncoder

data=pd.read\_csv('updated\_pollution\_dataset.csv')

data

**The data after executing the code:**

**# Encode target (Air Quality) into numeric values**

le = LabelEncoder()

data["Air Quality"] = le.fit\_transform(data["Air Quality"])

**# Features and Target**

X = data.drop(columns=["Air Quality"])

y = data["Air Quality"]

**# Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**# Bat Algorithm Parameters**

num\_bats = 10

max\_iter =20

loudness = 0.6

pulse\_rate = 0.5

frequency\_min, frequency\_max = 0, 2

**# Fitness Function with Additional Metrics**

def fitness\_function\_with\_metrics(solution):

selected\_features = np.where(solution > 0.5)[0]

if len(selected\_features) == 0:

return 0, 0, 0, 0 # Return 0 for accuracy, precision, recall, and F1 if no features are selected

**# Train model on selected features**

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train.iloc[:, selected\_features], y\_train)

preds = clf.predict(X\_test.iloc[:, selected\_features])

**# Calculate metrics**

accuracy = accuracy\_score(y\_test, preds)

precision = precision\_score(y\_test, preds, average='macro')

recall = recall\_score(y\_test, preds, average='macro')

f1 = f1\_score(y\_test, preds, average='macro')

return accuracy, precision, recall, f1

**# Initialize bats**

positions = np.random.rand(num\_bats, X.shape[1]) # Random positions in [0,1]

velocities = np.random.uniform(-1, 1, (num\_bats, X.shape[1])) # Random velocities

best\_global = positions[np.random.randint(0, num\_bats)] # Randomly select a bat

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = fitness\_function\_with\_metrics(best\_global)

**# Bat Algorithm**

for t in range(max\_iter):

for i in range(num\_bats):

**# Calculate frequency and update velocity and position**

frequency = frequency\_min + (frequency\_max - frequency\_min) \* np.random.rand()

velocities[i] += (positions[i] - best\_global) \* frequency

positions[i] = np.clip(positions[i] + velocities[i], 0, 1) # Ensure positions stay in range

**# Local search**

if np.random.rand() > pulse\_rate:

positions[i] = np.clip(best\_global + np.random.normal(0, 0.1, size=X.shape[1]), 0, 1)

**# Evaluate fitness**

accuracy, precision, recall, f1 = fitness\_function\_with\_metrics(positions[i])

if accuracy > best\_global\_accuracy and np.random.rand() < loudness:

best\_global = positions[i]

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = accuracy, precision, recall, f1

**# Update loudness and pulse rate**

loudness = max(0.1, loudness \* 0.95)

pulse\_rate = min(1.0, pulse\_rate \* 1.05)

**# Print progress**

print(f"Iteration {t + 1}:")

print(f" Best Accuracy = {best\_global\_accuracy:.4f}")

print(f" Best Precision = {best\_global\_precision:.4f}")

print(f" Best Recall = {best\_global\_recall:.4f}")

print(f" Best F1 Score = {best\_global\_f1:.4f}")

selected\_features = np.where(best\_global > 0.5)[0]

print(" Selected Features:", X.columns[selected\_features].tolist())

print()

**# Final Results**

print("Final Results:")

print(f"Best Accuracy: {best\_global\_accuracy:.4f}")

print(f"Best Precision: {best\_global\_precision:.4f}")

print(f"Best Recall: {best\_global\_recall:.4f}")

print(f"Best F1 Score: {best\_global\_f1:.4f}")

print("Selected Features:", X.columns[selected\_features].tolist())

**OUTPUT:**

Iteration 1:

Best Accuracy = 0.9273

Best Precision = 0.8990

Best Recall = 0.8786

Best F1 Score = 0.8867

Selected Features: ['PM2.5', 'PM10', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 2:

Best Accuracy = 0.9367

Best Precision = 0.9117

Best Recall = 0.9005

Best F1 Score = 0.9055

Selected Features: ['PM2.5', 'PM10', 'NO2', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 3:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM2.5', 'PM10', 'NO2', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 4:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM2.5', 'PM10', 'NO2', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 5:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 6:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Proximity\_to\_Industrial\_Areas', 'Population\_Density']

Iteration 7:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 8:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 9:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 10:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 11:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 12:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 13:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 14:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 15:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 16:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 17:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 18:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 19:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

Iteration 20:

Best Accuracy = 0.9420

Best Precision = 0.9235

Best Recall = 0.9049

Best F1 Score = 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

**Final Results:**

Best Accuracy: 0.9420

Best Precision: 0.9235

Best Recall: 0.9049

Best F1 Score: 0.9123

Selected Features: ['Humidity', 'PM10', 'NO2', 'CO', 'Population\_Density']

# **latest covid-19**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

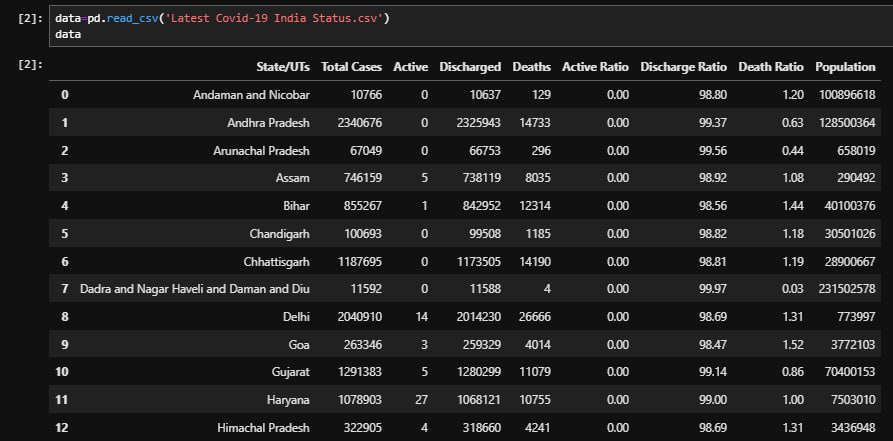
from sklearn.datasets import make\_classification

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

data=pd.read\_csv('Latest Covid-19 India Status.csv')

data

**The data after executing the code:**

**# Classify the Active Radio into two categories (High, Low)**

data['Death Ratio'] = (data['Death Ratio'] > 0.5).astype(int)

**# Features and Target**

X = data.drop(columns=["Death Ratio","State/UTs"])

y = data["Death Ratio"]

**# Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**# Bat Algorithm Parameters**

num\_bats = 10

max\_iter =20

loudness = 0.6

pulse\_rate = 0.5

frequency\_min, frequency\_max = 0, 2

**# Fitness Function with Additional Metrics**

def fitness\_function\_with\_metrics(solution):

selected\_features = np.where(solution > 0.5)[0]

if len(selected\_features) == 0:

return 0, 0, 0, 0 # Return 0 for accuracy, precision, recall, and F1 if no features are selected

**# Train model on selected features**

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train.iloc[:, selected\_features], y\_train)

preds = clf.predict(X\_test.iloc[:, selected\_features])

**# Calculate metrics**

accuracy = accuracy\_score(y\_test, preds)

precision = precision\_score(y\_test, preds)

recall = recall\_score(y\_test, preds)

f1 = f1\_score(y\_test, preds)

return accuracy, precision, recall, f1

**# Initialize bats**

positions = np.random.rand(num\_bats, X.shape[1]) # Random positions in [0,1]

velocities = np.random.uniform(-1, 1, (num\_bats, X.shape[1])) # Random velocities

best\_global = positions[np.random.randint(0, num\_bats)] # Randomly select a bat

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = fitness\_function\_with\_metrics(best\_global)

**# Bat Algorithm**

for t in range(max\_iter):

for i in range(num\_bats):

**# Calculate frequency and update velocity and position**

frequency = frequency\_min + (frequency\_max - frequency\_min) \* np.random.rand()

velocities[i] += (positions[i] - best\_global) \* frequency

positions[i] = np.clip(positions[i] + velocities[i], 0, 1) # Ensure positions stay in range

**# Local search**

if np.random.rand() > pulse\_rate:

positions[i] = np.clip(best\_global + np.random.normal(0, 0.1, size=X.shape[1]), 0, 1)

**# Evaluate fitness**

accuracy, precision, recall, f1 = fitness\_function\_with\_metrics(positions[i])

if accuracy > best\_global\_accuracy and np.random.rand() < loudness:

best\_global = positions[i]

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = accuracy, precision, recall, f1

# Update loudness and pulse rate

loudness = max(0.1, loudness \* 0.95)

pulse\_rate = min(1.0, pulse\_rate \* 1.05)

**# Print progress**

print(f"Iteration {t + 1}:")

print(f" Best Accuracy = {best\_global\_accuracy:.4f}")

print(f" Best Precision = {best\_global\_precision:.4f}")

print(f" Best Recall = {best\_global\_recall:.4f}")

print(f" Best F1 Score = {best\_global\_f1:.4f}")

selected\_features = np.where(best\_global > 0.5)[0]

print(" Selected Features:", X.columns[selected\_features].tolist())

print()

**# Final Results**

print("Final Results:")

print(f"Best Accuracy: {best\_global\_accuracy:.4f}")

print(f"Best Precision: {best\_global\_precision:.4f}")

print(f"Best Recall: {best\_global\_recall:.4f}")

print(f"Best F1 Score: {best\_global\_f1:.4f}")

print("Selected Features:", X.columns[selected\_features].tolist())

**OUTPUT:**

Iteration 1:

Best Accuracy = 0.9091

Best Precision = 0.9091

Best Recall = 1.0000

Best F1 Score = 0.9524

Selected Features: ['Total Cases', 'Active', 'Deaths', 'Discharge Ratio']

Iteration 2:

Best Accuracy = 0.9091

Best Precision = 0.9091

Best Recall = 1.0000

Best F1 Score = 0.9524

Selected Features: ['Total Cases', 'Active', 'Deaths', 'Discharge Ratio']

Iteration 3:

Best Accuracy = 0.9091

Best Precision = 0.9091

Best Recall = 1.0000

Best F1 Score = 0.9524

Selected Features: ['Total Cases', 'Active', 'Deaths', 'Discharge Ratio']

Iteration 4:

Best Accuracy = 0.9091

Best Precision = 0.9091

Best Recall = 1.0000

Best F1 Score = 0.9524

Selected Features: ['Total Cases', 'Active', 'Deaths', 'Discharge Ratio']

Iteration 5:

Best Accuracy = 0.9091

Best Precision = 0.9091

Best Recall = 1.0000

Best F1 Score = 0.9524

Selected Features: ['Total Cases', 'Active', 'Deaths', 'Discharge Ratio']

Iteration 6:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 7:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 8:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 9:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 10:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 11:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 12:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 13:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 14:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 15:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 16:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 17:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 18:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 19:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

Iteration 20:

Best Accuracy = 1.0000

Best Precision = 1.0000

Best Recall = 1.0000

Best F1 Score = 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

**Final Results:**

Best Accuracy: 1.0000

Best Precision: 1.0000

Best Recall: 1.0000

Best F1 Score: 1.0000

Selected Features: ['Total Cases', 'Discharged', 'Discharge Ratio', 'Population']

# **customer churn**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_classification

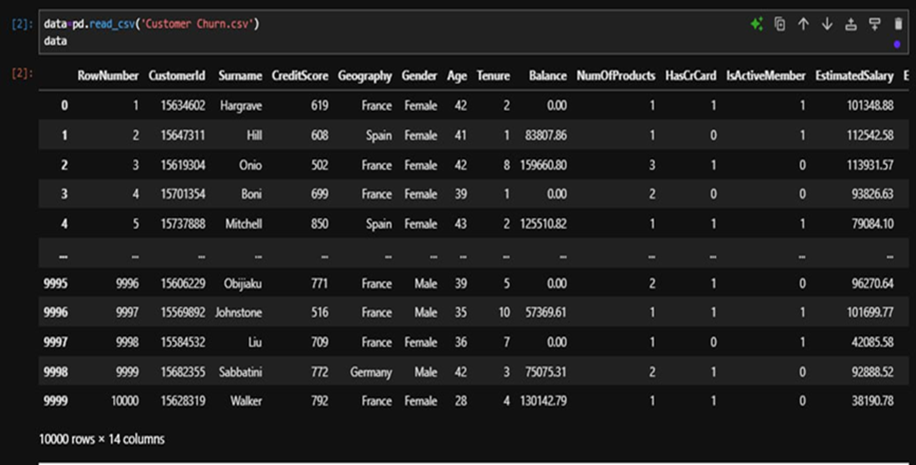
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

data=pd.read\_csv('Customer Churn.csv')

data

**The data after executing the code:**



**# Features and Target**

X = data.drop(columns=["Exited","Surname","Geography","Gender"])

y = data["Exited"]

**# Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**# Bat Algorithm Parameters**

num\_bats = 10

max\_iter =20

loudness = 0.6

pulse\_rate = 0.5

frequency\_min, frequency\_max = 0, 2

**# Fitness Function with Additional Metrics**

def fitness\_function\_with\_metrics(solution):

selected\_features = np.where(solution > 0.5)[0]

if len(selected\_features) == 0:

return 0, 0, 0, 0 # Return 0 for accuracy, precision, recall, and F1 if no features are selected

**# Train model on selected features**

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train.iloc[:, selected\_features], y\_train)

preds = clf.predict(X\_test.iloc[:, selected\_features])

**# Calculate metrics**

accuracy = accuracy\_score(y\_test, preds)

precision = precision\_score(y\_test, preds)

recall = recall\_score(y\_test, preds)

f1 = f1\_score(y\_test, preds)

return accuracy, precision, recall, f1

**# Initialize bats**

positions = np.random.rand(num\_bats, X.shape[1]) # Random positions in [0,1]

velocities = np.random.uniform(-1, 1, (num\_bats, X.shape[1])) # Random velocities

best\_global = positions[np.random.randint(0, num\_bats)] # Randomly select a bat

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = fitness\_function\_with\_metrics(best\_global)

**# Bat Algorithm**

for t in range(max\_iter):

for i in range(num\_bats):

# Calculate frequency and update velocity and position

frequency = frequency\_min + (frequency\_max - frequency\_min) \* np.random.rand()

velocities[i] += (positions[i] - best\_global) \* frequency

positions[i] = np.clip(positions[i] + velocities[i], 0, 1) # Ensure positions stay in range

**# Local search**

if np.random.rand() > pulse\_rate:

positions[i] = np.clip(best\_global + np.random.normal(0, 0.1, size=X.shape[1]), 0, 1)

**# Evaluate fitness**

accuracy, precision, recall, f1 = fitness\_function\_with\_metrics(positions[i])

if accuracy > best\_global\_accuracy and np.random.rand() < loudness:

best\_global = positions[i]

best\_global\_accuracy, best\_global\_precision, best\_global\_recall, best\_global\_f1 = accuracy, precision, recall, f1

**# Update loudness and pulse rate**

loudness = max(0.1, loudness \* 0.95)

pulse\_rate = min(1.0, pulse\_rate \* 1.05)

**# Print progress**

print(f"Iteration {t + 1}:")

print(f" Best Accuracy = {best\_global\_accuracy:.4f}")

print(f" Best Precision = {best\_global\_precision:.4f}")

print(f" Best Recall = {best\_global\_recall:.4f}")

print(f" Best F1 Score = {best\_global\_f1:.4f}")

selected\_features = np.where(best\_global > 0.5)[0]

print(" Selected Features:", X.columns[selected\_features].tolist())

print()

**# Final Results**

print("Final Results:")

print(f"Best Accuracy: {best\_global\_accuracy:.4f}")

print(f"Best Precision: {best\_global\_precision:.4f}")

print(f"Best Recall: {best\_global\_recall:.4f}")

print(f"Best F1 Score: {best\_global\_f1:.4f}")

print("Selected Features:", X.columns[selected\_features].tolist())

OUTPUT:

Iteration 1:

Best Accuracy = 0.8150

Best Precision = 0.5381

Best Recall = 0.3510

Best F1 Score = 0.4249

Selected Features: ['CreditScore', 'Age', 'Balance', 'IsActiveMember']

Iteration 2:

Best Accuracy = 0.8150

Best Precision = 0.5381

Best Recall = 0.3510

Best F1 Score = 0.4249

Selected Features: ['CreditScore', 'Age', 'Balance', 'IsActiveMember']

Iteration 3:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'CreditScore', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember', 'EstimatedSalary']

Iteration 4:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'CreditScore', 'Age', 'NumOfProducts', 'IsActiveMember', 'EstimatedSalary']

Iteration 5:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'CreditScore', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 6:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'CreditScore', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 7:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'CreditScore', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 8:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 9:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 10:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 11:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 12:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 13:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 14:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 15:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 16:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 17:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 18:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 19:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

Iteration 20:

Best Accuracy = 0.8593

Best Precision = 0.7411

Best Recall = 0.4264

Best F1 Score = 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

**Final Results:**

Best Accuracy: 0.8593

Best Precision: 0.7411

Best Recall: 0.4264

Best F1 Score: 0.5413

Selected Features: ['CustomerId', 'Age', 'NumOfProducts', 'EstimatedSalary']

# **Code steps followed**

1. Data Preprocessing

* Load the **heart.csv** dataset.
* Split the dataset into features (X) and the target variable (y).
* Perform a train-test split to prepare data for model evaluation.

2.DefineFitness Function

* The **fitness\_function** evaluates a solution's quality by:
* Selecting features based on a threshold.
* Training a Random Forest classifier on the selected features.
* Returning the accuracy on the test data.

3. Initialize Parameters

* **Positions:** Random initial feature selection values for each bat.
* **Velocities:** Random initial movement directions.
* **Global Best:** A randomly selected initial solution with its fitness.

4. Main Bat Algorithm Loop

● For each iteration:

1. Update frequency, velocity, and position for each bat.
2. Perform a local random search with a probability based on pulse rate.
3. Evaluate the fitness of the new positions.
4. Update the global best solution if a better fitness is found.

● Adjust loudness and pulse rate to balance exploration and exploitation:

5. Output Results

* After all iterations, the algorithm outputs the best fitness (accuracy) and the selected features.

Key Parameters

* **num\_bats:** Number of bats in the population.
* **max\_iter:** Maximum number of iterations.
* **loudness &pulse\_rate:** Control exploration and exploitation.
* **frequency\_min & frequency\_max:** Range for frequency values.

# **Conclusion**

This implementation of the Bat Algorithm demonstrates its capability for feature selection by identifying the most relevant features that enhance a Random Forest classifier's accuracy. The iterative optimization process balances global search and local refinement to find the optimal feature subset.