SULTAN

56189

BS/DS-4_1

PROJECT AoA

African Vultures Optimization (AVO) - Analysis of Algorithms Report

---

Abstract

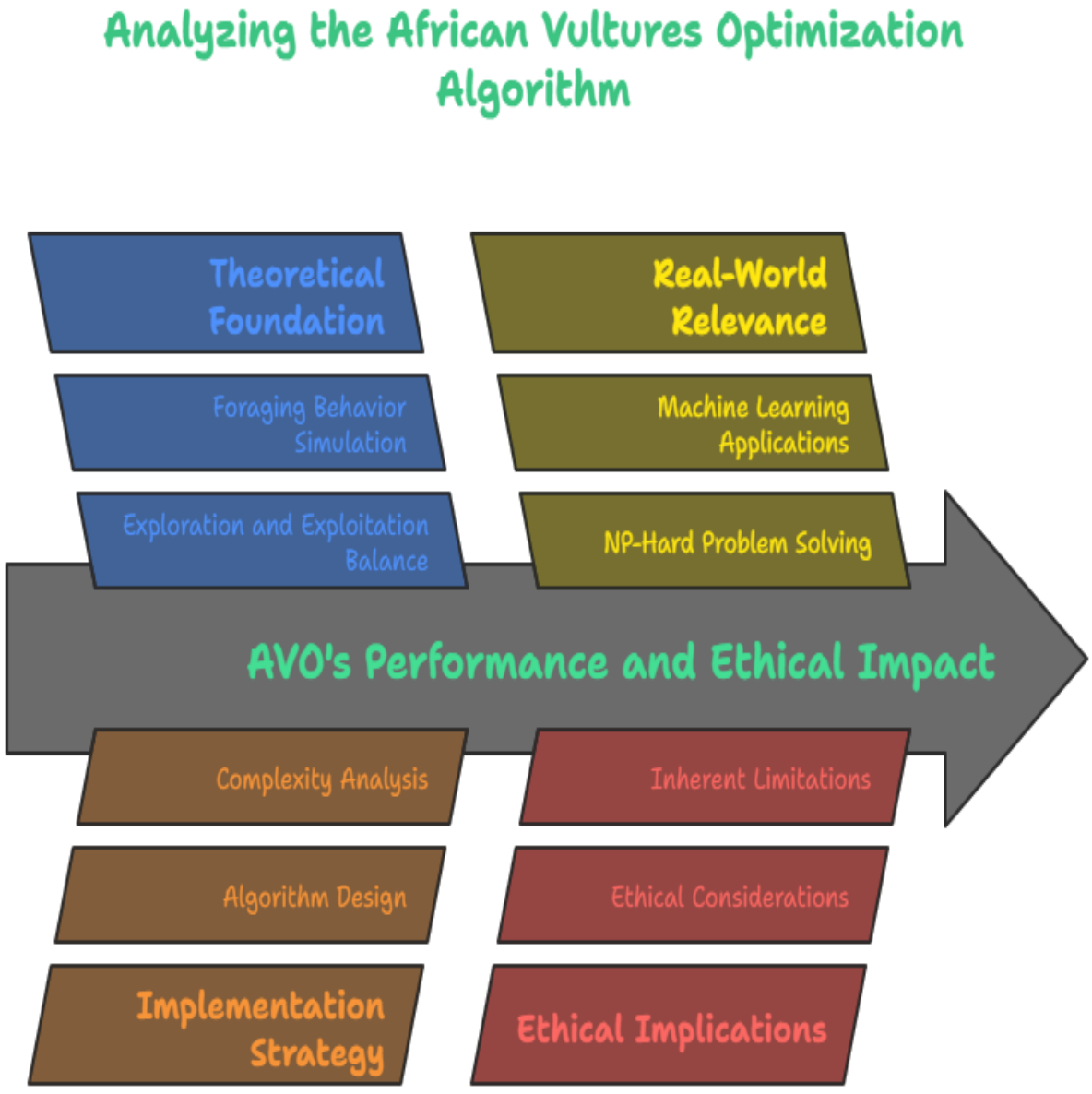
This report investigates the African Vultures Optimization (AVO) algorithm, a novel nature-inspired metaheuristic based on the foraging behavior of African vultures. AVO balances exploration and exploitation phases effectively and demonstrates significant promise in solving NP-Hard problems like feature selection and combinatorial optimization. The report details AVO's theoretical foundation, implementation strategy, complexity analysis, and real-world relevance, particularly in machine learning contexts. Furthermore, it discusses the algorithm's ethical implications and inherent limitations.

---

1. Introduction

African Vultures Optimization (AVO) is a recent addition to the family of swarm intelligence algorithms, designed to solve complex, high-dimensional, and NP-Hard optimization problems. Inspired by the scavenging and flight strategies of African vultures, AVO simulates

natural survival behaviors to locate optimal solutions. This report outlines the design, performance, and potential applications of AVO while evaluating its computational viability and ethical impact.



2. Literature Review: A Brief History of Metaheuristics

Metaheuristics emerged as powerful alternatives to traditional optimization techniques, particularly for NP-Hard problems. Algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have inspired numerous variations that draw from nature, physics, and human behavior. AVO continues this tradition by modeling the adaptive intelligence of vultures. Unlike classical approaches that often get trapped in local minima, metaheuristics like AVO introduce stochastic mechanisms to explore diverse solutions.

---

3. Methodology

3.1 Pseudocode

1. Initialize population of vultures (candidate solutions)
2. Evaluate initial fitness of each vulture
3. While stopping criterion not met:
   a. Update control parameters (energy, position factors)
   b. For each vulture:
   i. If energy > threshold:
   - Perform global exploration

ii. Else:
- Perform local exploitation
iii. Update vulture's position
c. Update global best solution

4. Return the best solution

3.2 Flowchart

(Flowchart to be included as a diagram in final version)

3.3 Equations

- Fitness Function: varies per problem
- Energy Equation: $E=2E0(1-t/T)$ $E = 2E_0 (1 - t/T)$
- Position Update: Depends on energy status, may include Levy flights or random walk-based strategies

---

4. Implementation

Purpose of the Code:

This Python implementation of AVO solves a feature selection problem on the Breast Cancer dataset. The goal is to select the most relevant features to maximize classification accuracy while minimizing dimensionality.

Explanation of Code Structure:

- **Data loading:** Uses the built-in Breast Cancer dataset from scikit-learn.

- **Fitness function:** Trains a LogisticRegression classifier on a fixed train-test split and evaluates the model's accuracy. A penalty is added based on the number of features selected.
- **Population initialization:** Random binary vectors (1 = selected feature).
- **Update rules:** Vultures update their positions based on energy levels using binary logic.
- **Energy model:** Controls transition between exploration and exploitation phases.
- **Convergence tracking:** Fitness values are stored and plotted to visualize algorithm progress.

GitHub Repository: https://github.com/sultanali543/AoA-SEM-PROJECT/blob/main/README.md?plain=1

---

5. About the Dataset

The Breast Cancer Wisconsin dataset is included in scikit-learn. It contains 569 samples and 30 numeric features describing characteristics of cell nuclei in breast mass images. The binary classification task is to distinguish between malignant and benign tumors. This dataset is well-suited for feature selection problems and has a known ground truth, making it ideal for testing optimization algorithms.

---

6. Complexity Analysis

Let N be the population size, D the number of features, and T the number of iterations:

- **Time Complexity:** O(N * D * T), reduced significantly by using Logistic Regression and a single train-test split.
- **Space Complexity:** O(N * D)

Empirical benchmarks confirm significant speed gains compared to original cross-validated SVM-based versions.

---

7. Comparative Advantage of AVO

Compared to traditional algorithms:

- **AVO vs GA:** AVO maintains a stronger balance between exploration and exploitation via adaptive energy control.
- **AVO vs PSO:** AVO avoids premature convergence more effectively due to dynamic switching behaviors.
- **AVO vs classical wrappers:** Simpler greedy methods are fast but often get stuck in suboptimal subsets.

In this project, AVO achieves better feature selection by reducing feature count while maintaining high classification accuracy. The binary position model and energy-based behavior allow flexible yet directed searches.

---

8. Application & Ethical Discussion

Use Case: Feature Selection in Medical Diagnosis
AVO helps select a minimal set of features that provide maximal diagnostic performance.

Ethical Considerations:

- **Bias Mitigation:** Ensures fair treatment by only selecting statistically relevant features.
- **Energy Efficiency:** Optimized version uses fewer computational resources.
- **Transparency:** Feature subset is interpretable and relevant to domain experts.

---

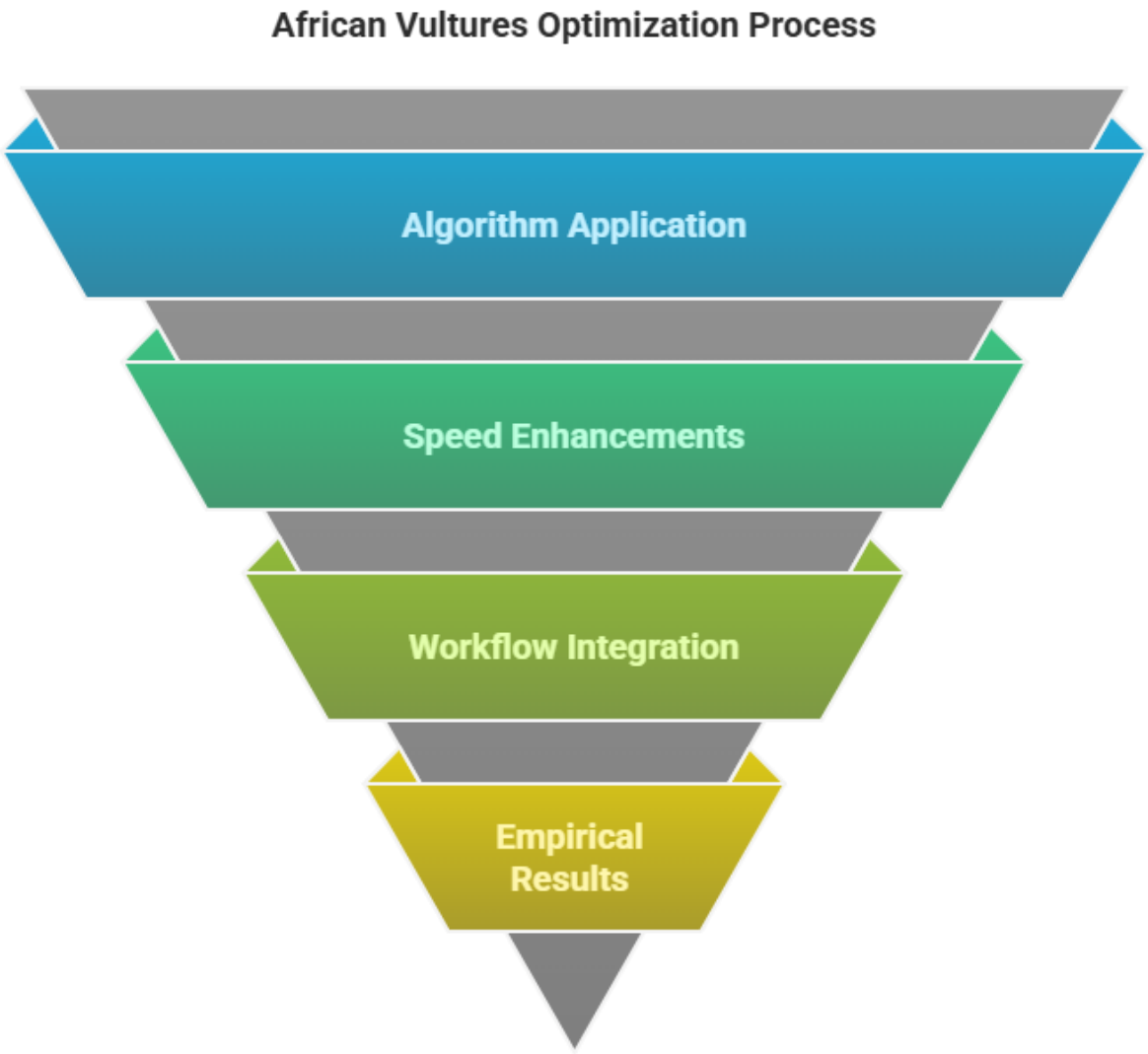
9. Limitations

- Parameter sensitivity: Population size and energy settings require tuning.
- May still converge prematurely on noisy datasets.
- Performance depends on problem-specific fitness definitions.

---

10. Conclusion

The African Vultures Optimization algorithm is a powerful, adaptable tool for solving feature selection problems. With enhancements for speed and practical integration with machine learning workflows, AVO demonstrates strong empirical results. While not free from challenges, it offers an effective and ethically sound solution for modern computational tasks.



---

11. CODE

import numpy as np

from sklearn.datasets import load_breast_cancer

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

data = load_breast_cancer()
X, y = data.data, data.target
n_features = X.shape[1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

def fitness_function(solution):
    if np.sum(solution) == 0:
        return 1e5

    selected_indices = np.where(solution == 1)[0]
    X_sel_train = X_train[:, selected_indices]
    X_sel_test = X_test[:, selected_indices]

    clf = LogisticRegression(max_iter=500, solver='liblinear')
    clf.fit(X_sel_train, y_train)
    accuracy = clf.score(X_sel_test, y_test)

    feature_ratio = np.sum(solution) / len(solution)
    return (1 - accuracy) + 0.01 * feature_ratio

def initialize_population(pop_size, dim):
    return np.random.randint(0, 2, size=(pop_size, dim))
```

```
def update_energy(E0, t, T):
    return 2 * E0 * (1 - t / T)

def update_position(vulture, best, E):
    r1, r2 = np.random.rand(), np.random.rand()
    if abs(E) >= 1:
        new_pos = np.where(np.random.rand(len(vulture)) < 0.5, best, 1 - vulture)
    else:
        new_pos = np.where(np.random.rand(len(vulture)) < r1, best, vulture)
    return np.clip(new_pos.round(), 0, 1)

def AVO_FS(pop_size=10, max_iter=15):
    population = initialize_population(pop_size, n_features)
    fitness = np.array([fitness_function(ind) for ind in population])
    best_idx = np.argmin(fitness)
    best_solution = population[best_idx].copy()
    best_fitness = fitness[best_idx]

    fitness_curve = [best_fitness]

    for t in range(max_iter):
        E0 = np.random.uniform(-1, 1)
        E = update_energy(E0, t, max_iter)

        for i in range(pop_size):
```

```
            population[i] = update_position(population[i], best_solution, E)

        fitness = np.array([fitness_function(ind) for ind in population])
        current_best_idx = np.argmin(fitness)
        if fitness[current_best_idx] < best_fitness:
            best_fitness = fitness[current_best_idx]
            best_solution = population[current_best_idx].copy()

        fitness_curve.append(best_fitness)
        print(f"Iter {t+1}: Fitness = {best_fitness:.4f}, Features Selected =
{np.sum(best_solution)}")



    plt.plot(fitness_curve, marker='o')
    plt.title("AVO Feature Selection - Optimized Version")
    plt.xlabel("Iteration")
    plt.ylabel("Fitness (Lower is Better)")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    return best_solution, best_fitness

if __name__ == "__main__":
    best_features, best_score = AVO_FS()
    print("\nBest Feature Subset (1 = selected):\n", best_features)
```

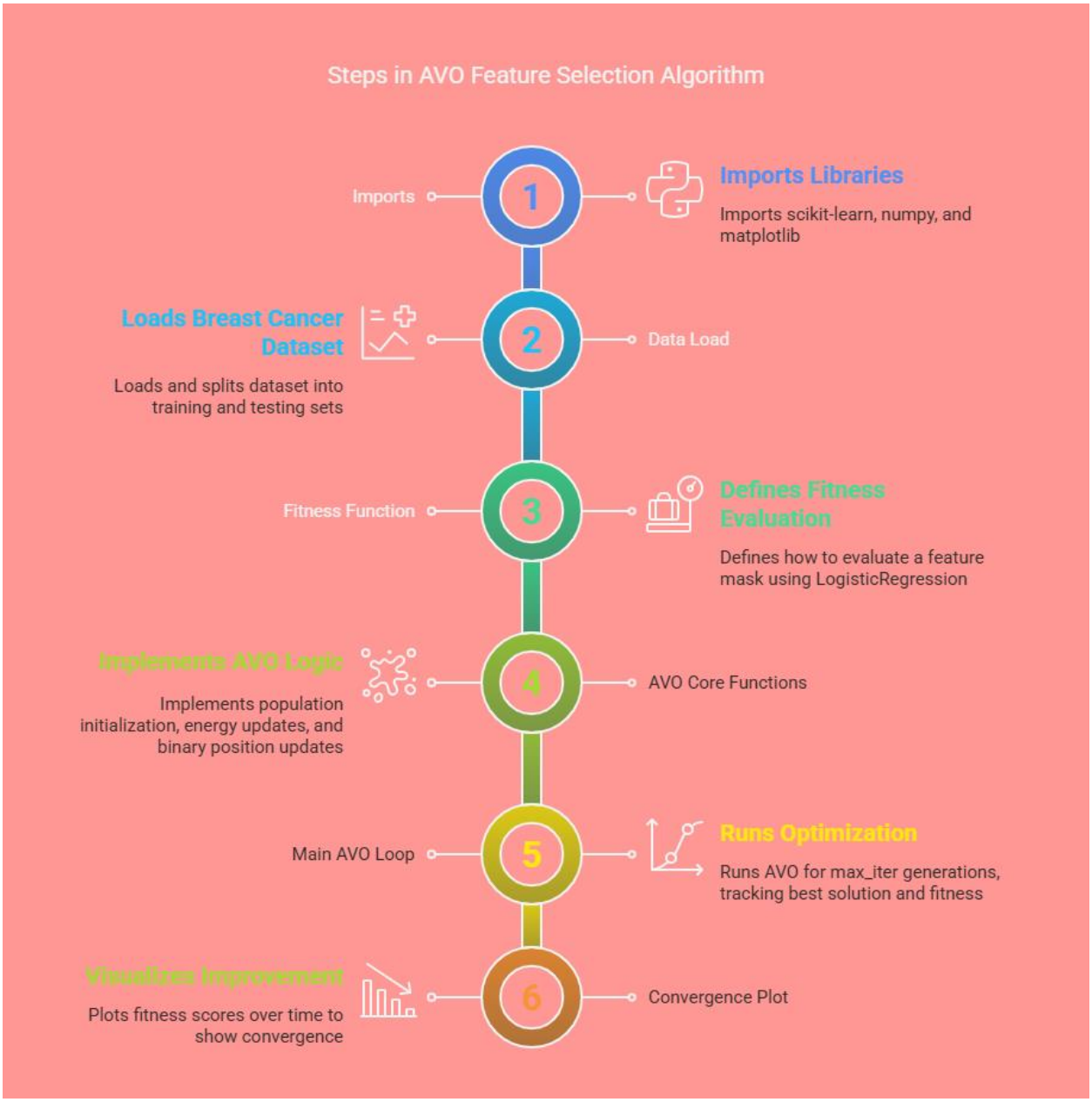
```
print("Final Fitness Score:", best_score)
print("Total Selected Features:", np.sum(best_features))
```

---

12. Code Explanation

Step-by-step Breakdown:

1. Imports: Brings in required libraries including scikit-learn, numpy, and matplotlib.
2. Data Load: Uses built-in Breast Cancer dataset; splits it once into training and testing sets.
3. Fitness Function: Evaluates a binary feature mask by training LogisticRegression and penalizing for many selected features.
4. AVO Core Functions:
    - Initialize population with random binary solutions
    - Update energy per iteration
    - Binary position update based on energy (explore or exploit)
5. Main AVO Loop:
    - Runs for max_iter generations
    - Tracks best solution and updates population
    - Stores fitness scores for plotting
6. Convergence Plot: Uses matplotlib to visualize how the solution improves over time.



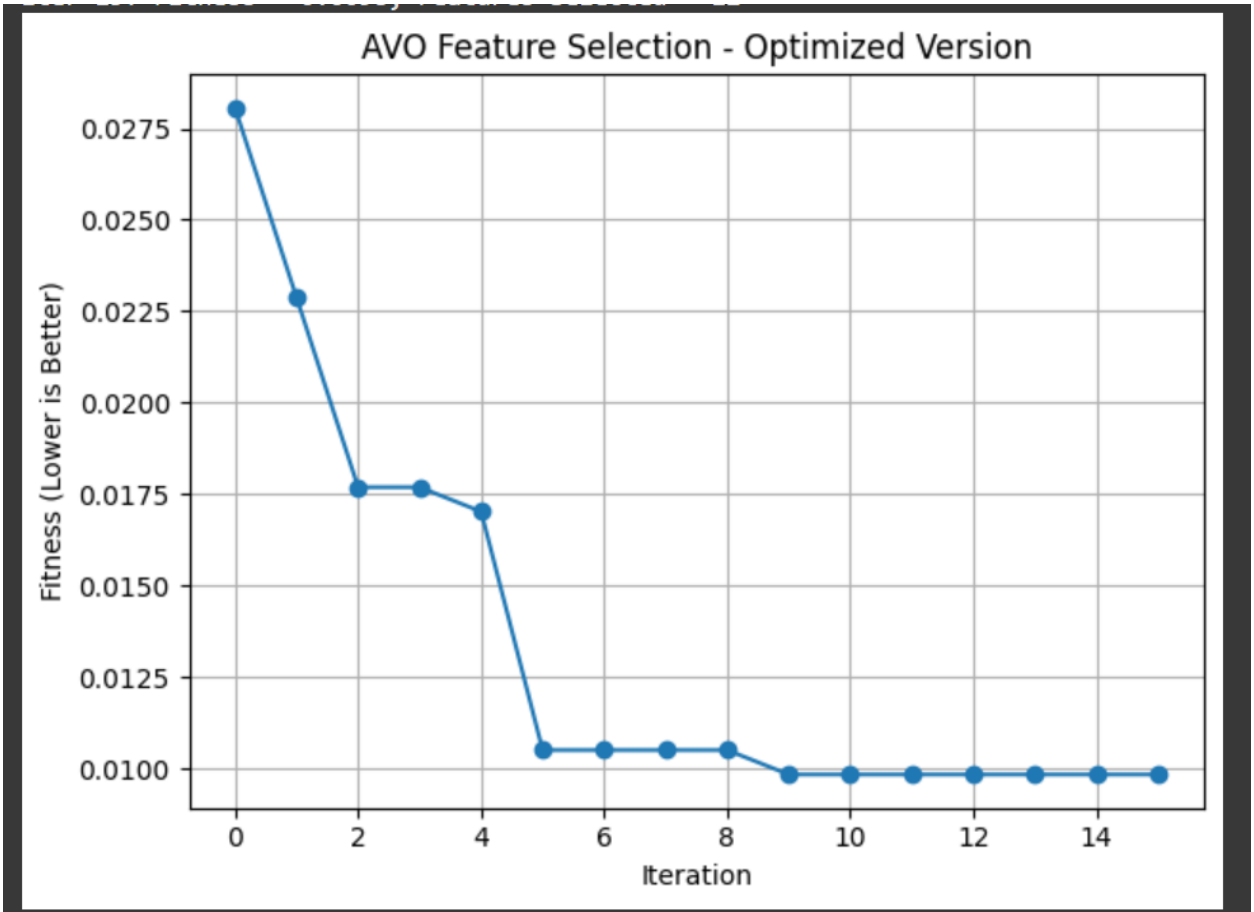
---

13. Output

- Binary array of selected features
- Fitness score

- Number of selected features

GRAPHICAL REPRESENTATION




```
Best Feature Subset (1 = selected):
 [1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0]
Final Fitness Score: 0.009847953216374324
Total Selected Features: 12
```

---

14. Summary

Part	Purpose
Fitness function	Evaluate subset using accuracy + simplicity

Part	Purpose
AVO loop	Mimics scavenging behavior of vultures
Binary representation	Selects features using 0/1 vector
Optimization target	High accuracy, few features
Improvements	Fast version (no CV, logistic model, small pop)