Particle Swarm Optimization

Under supervision of:
Dr. Othman Ali
Dr. Eslam Ezzat

Made by :

- Omar Roshdy Ahmed
- Fares Karim Foully
- Kareem Mohamed ElSayed
- Ali mohamed Ali
- Mohamed Salah Mohamed
- Mohamed Ahmed Mahmoud
- Eslam Mahmoud Mahmoud
- Abdelrahman Mohamed Mahfouz
- Mohamed Fathi Zaki

Introduction

This document provides a detailed explanation of a Python script that demonstrates data preprocessing, cleaning using Particle Swarm Optimization (PSO), and model building using Random Forest Classifier. The script is designed for handling customer churn data and evaluating model performance.

1. Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import accuracy_score,
```

Importing necessary libraries for data processing, modeling, and evaluation. Each library is crucial for specific tasks:

- **pandas**: For data manipulation and handling tabular data.
- **numpy**: For numerical operations and array handling.
- **sklearn.model_selection**: For splitting data into training and testing sets.
- **sklearn.ensemble**: For using the Random Forest Classifier.
- **sklearn.metrics**: For evaluating model performance using accuracy, F1 score, precision, and recall.

2. Loading Data

```
data = pd.read_csv('Customer Churn.csv')
```

The dataset is read from a CSV file named 'Customer Churn.csv' using pandas. This dataset contains information about customers and their churn status.

3. Fitness Function

```
def fitness_function(data):
    missing_values = data.isnull().sum().sum()
    duplicates = data.duplicated().sum()
    return missing_values + duplicates
```

The `fitness_function` calculates the cleanliness of the dataset by identifying missing values and duplicate rows. The goal is to minimize this function for better data quality.

1. Steps:
    - Count the total number of missing values.
    - Count the total number of duplicate rows.
    - Return the sum of missing values and duplicates.

4. Particle Swarm Optimization (PSO)

PSO is a metaheuristic algorithm inspired by the behavior of bird flocks. It is used here to clean the dataset by iteratively improving the dataset's fitness.

1. Key steps in PSO:

a) Initialize multiple copies of the dataset (particles).

```
particles = [data.copy() for _ in range(n_particles)]

p_best = particles.copy()
```

b) Evaluate the fitness of each particle.

```
p_best_fitness = [fitness_function(p) for p in particles]
```

c) Iteratively improve the particles by removing random rows and filling missing values.

```
        for i, particle in enumerate(particles):

            if np.random.rand() > 0.5:

                particle = particle.drop(particle.sample(1).index)

            for col in
particle.select_dtypes(include='number').columns:

                particle[col] =
particle[col].fillna(particle[col].mean())

            for col in
particle.select_dtypes(exclude='number').columns:

                particle[col] =
particle[col].fillna(particle[col].mode()[0])
```

d) Update the best individual (pBest) and global (gBest) solutions.

```
            if fitness < p_best_fitness[i]:

                p_best[i] = particle

                p_best_fitness[i] = fitness
```

```
            if fitness < fitness_function(g_best):
                g_best = particle
```

5. Data Preprocessing

Preprocessing steps include:

1. Converting categorical variables to numeric using one-hot encoding.
2. Splitting data into independent variables (X) and target variable (y).
3. Dividing data into training and testing sets using an 80-20 split.

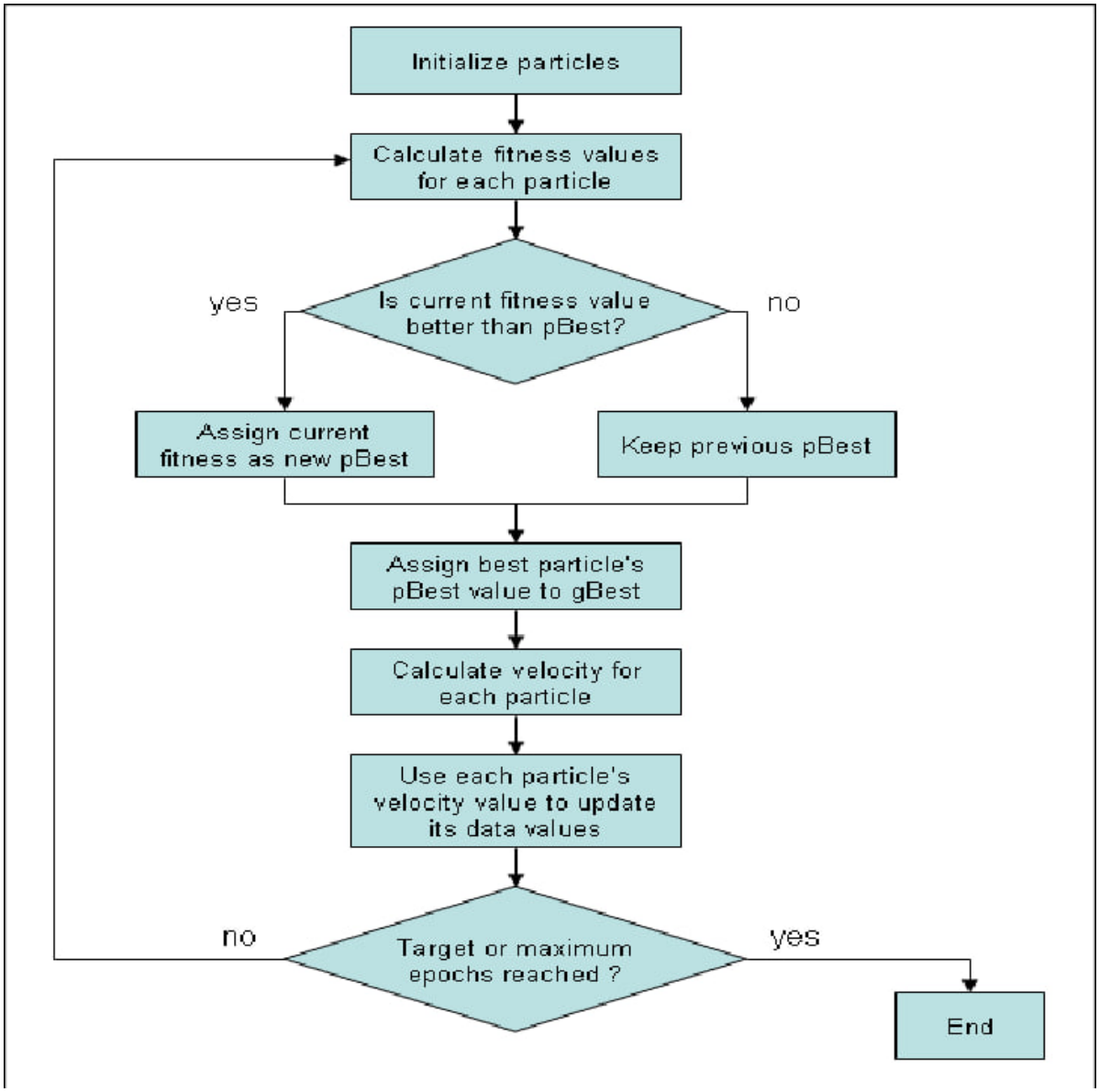
6. Model Training and Evaluation

The script trains a Random Forest Classifier on the training dataset. Model performance is evaluated on the testing dataset using the following metrics:

- **Accuracy**: Overall correctness of the model.
- **F1 Score**: Harmonic mean of precision and recall.
- **Precision**: Correctness of positive predictions.
- **Recall**: Ability to identify positive cases.

7. Results

The script prints the evaluation metrics after testing the model, providing insights into the model's predictive capabilities and areas for improvement.



Conclusion

This script demonstrates the end-to-end process of data cleaning, preprocessing, and model building using Python libraries and machine learning techniques. It showcases how Particle Swarm Optimization can be adapted for data cleaning tasks.