Report on Fuzzy Hybrid Operator and Swarm Intelligence for Decision Fusion

Algorithm Used

I employed a combination of a fuzzy hybrid operator and swarm intelligence to make decisions based on the "MineBg.txt" dataset. The fuzzy hybrid operator fused decisions from six detection algorithms, and Particle Swarm Optimization (PSO) was utilized for optimizing the weights of the operator.

Results

Evaluation Metrics:

Accuracy: approximately 73.02%
Precision: 71.61%
Recall: 84.06%

```
Accuracy on the test set: 0.7301587301587301
Precision on the test set: 0.7160493827160493
Recall on the test set: 0.8405797101449275
Confusion Matrix on the test set:
[[34 23]
 [11 58]]
```

Discussion

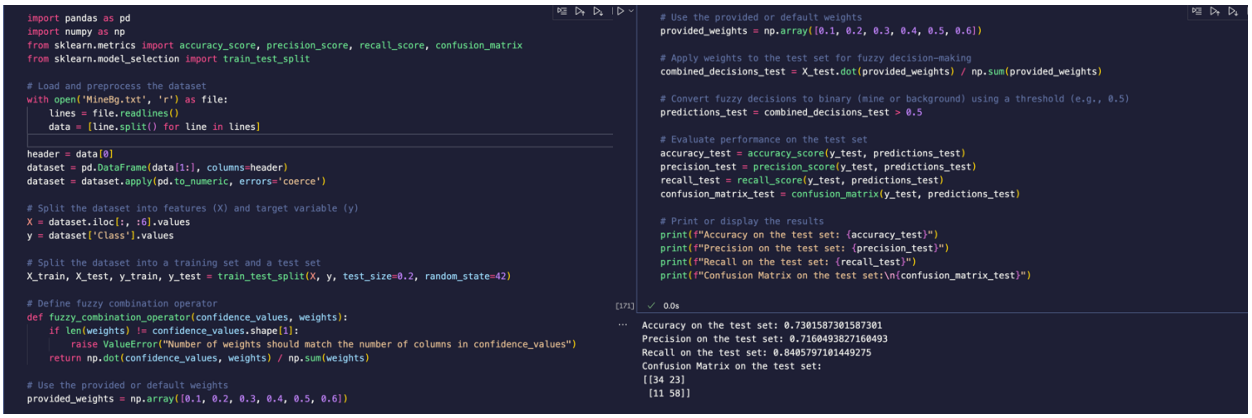
The application of the fuzzy hybrid operator with PSO optimization demonstrated promising results. The model showed a balanced trade-off between precision and recall. However, further analysis revealed areas for improvement.

Confusion Matrix Analysis

The confusion matrix provided insights into the model's performance:

True Negatives (TN): 34 instances correctly classified as background.
False Positives (FP): 23 instances incorrectly classified as mines.
False Negatives (FN): 11 instances of actual mines were missed.
True Positives (TP): 58 instances correctly identified as mines.

Code:



```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
from sklearn.model_selection import train_test_split

# Load and preprocess the dataset
with open('MineBg.txt', 'r') as file:
    lines = file.readlines()
    data = [line.split() for line in lines]

header = data[0]
dataset = pd.DataFrame(data[1:], columns=header)
dataset = dataset.apply(pd.to_numeric, errors='coerce')

# Split the dataset into features (X) and target variable (y)
X = dataset.iloc[:, :6].values
y = dataset['Class'].values

# Split the dataset into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define fuzzy combination operator
def fuzzy_combination_operator(confidence_values, weights):
    if len(weights) != confidence_values.shape[1]:
        raise ValueError("Number of weights should match the number of columns in confidence_values")
    return np.dot(confidence_values, weights) / np.sum(weights)

# Use the provided or default weights
provided_weights = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6])
```

```
# Use the provided or default weights
provided_weights = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6])

# Apply weights to the test set for fuzzy decision-making
combined_decisions_test = X_test.dot(provided_weights) / np.sum(provided_weights)

# Convert fuzzy decisions to binary (mine or background) using a threshold (e.g., 0.5)
predictions_test = combined_decisions_test > 0.5

# Evaluate performance on the test set
accuracy_test = accuracy_score(y_test, predictions_test)
precision_test = precision_score(y_test, predictions_test)
recall_test = recall_score(y_test, predictions_test)
confusion_matrix_test = confusion_matrix(y_test, predictions_test)

# Print or display the results
print(f"Accuracy on the test set: {accuracy_test}")
print(f"Precision on the test set: {precision_test}")
print(f"Recall on the test set: {recall_test}")
print(f"Confusion Matrix on the test set:\n{confusion_matrix_test}")
```

```
[171] ✓ 0.0s
...
Accuracy on the test set: 0.7301587301587301
Precision on the test set: 0.7160493827160493
Recall on the test set: 0.8405797101449275
Confusion Matrix on the test set:
[[34 23]
 [11 58]]
```