```
import numpy as np

# Generate dummy data for demonstration purposes (replace this with
your actual data)
def generate_dummy_data(samples=100, features=10):
    data = np.random.rand(samples, features)
    labels = np.random.randint(0, 2, size=samples)  # Binary labels (0
or 1)
    return data, labels

# Define the AIRS algorithm (Artificial Immune Recognition System)
class AIRS:
    def __init__(self, num_detectors=10):
        self.num_detectors = num_detectors

    def train(self, X, y):
        # Randomly initialize detectors from the training data
        self.detectors = X[np.random.choice(len(X),
self.num_detectors, replace=False)]
        self.detector_labels = y[np.random.choice(len(X),
self.num_detectors, replace=False)]  # Store the labels for detectors

    def predict(self, X):
        predictions = []
        for sample in X:
            # Calculate the Euclidean distance between the sample and
each detector
            distances = np.linalg.norm(self.detectors - sample,
axis=1)
            # Find the label of the closest detector
            closest_detector_index = np.argmin(distances)

predictions.append(self.detector_labels[closest_detector_index])
        return predictions

# Generate dummy data
data, labels = generate_dummy_data()

# Split data into training and testing sets
split_ratio = 0.8
split_index = int(split_ratio * len(data))
train_data, test_data = data[:split_index], data[split_index:]
train_labels, test_labels = labels[:split_index], labels[split_index:]

# Initialize and train the AIRS model
airs = AIRS(num_detectors=10)
airs.train(train_data, train_labels)

# Test AIRS on the test set
predictions = airs.predict(test_data)
```

```
# Evaluate accuracy
accuracy = np.mean(predictions == test_labels)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8
```