ASSIGNMENT No: 7

```python
import numpy as np

# Define the AIRS algorithm
class AIRS:
    def __init__(self, num_detectors=10, hypermutation_rate=0.1):
        self.num_detectors = num_detectors
        self.hypermutation_rate = hypermutation_rate
        self.detectors = None
        self.detector_labels = None

    def train(self, X, y):
        indices = np.random.choice(len(X), self.num_detectors, replace=False)
        self.detectors = X[indices]
        self.detector_labels = y[indices]  # Store labels of selected detectors

    def predict(self, X):
        predictions = []
        for sample in X:
            distances = np.linalg.norm(self.detectors - sample, axis=1)
            nearest_index = np.argmin(distances)
            prediction = self.detector_labels[nearest_index]  # Use label of closest detector
            predictions.append(prediction)
        return np.array(predictions)

# Generate dummy data
def generate_dummy_data(samples=100, features=10):
    data = np.random.rand(samples, features)
    labels = np.random.randint(0, 2, size=samples)  # Binary classification (0 or 1)
    return data, labels

# Generate 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 AIRS
airs = AIRS(num_detectors=10, hypermutation_rate=0.1)
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:.2f}")
```

```
Accuracy: 0.70
```