# Name: Nirmal Avhad

# Roll no: 71

# Subject: CL 3

**Practical No 7**

**Title :-** To apply the artificial immune pattern recognition to perform a task of structure damage Classification.

**Problem Statement:-** Develop an artificial immune pattern recognition system for the task of  structural damage classification.

**Objective :-**

1. Design a robust algorithm capable of identifying patterns indicative of structural damage  within complex datasets.

2. Implement a scalable solution that can accurately classify various types and extents of  structural damage with high efficiency.

**Outcome :-**

After completion of this assignment students are able to understand how to The developed system  will enable automated and accurate identification of structural damage, aiding in timely maintenance  and ensuring safety.

**Software Requirements:-**

• Python (3.x recommended)

• Jupyter Notebook or any Python IDE

**Hardware Requirement :-**

A machine with sufficient RAM and processing power for model training (8GB RAM  recommended)

**Prerequisities:-**

• Basic understanding of Java programming

**Theory :-**

**Introduction to AIS:**

• Briefly discuss the biological immune system and its key functions.

• Introduce the concept of AIS and its principles for pattern recognition.

• Explain the analogy between antigens (foreign invaders) and damage patterns, and antibodies  (immune cells) and damage detectors.

**2. Setting Up the AIS Model:**

• Divide the students into groups and assign each group a specific damage type  (e.g., crack, corrosion).

• Provide each group with a training dataset containing features of healthy and damaged structures  related to their assigned damage type.

• Guide students through the process of encoding the data into a format suitable for the AIS algorithm  (e.g., binary strings representing feature values).

• Introduce the chosen AIS algorithm and its key parameters (e.g., population size, mutation rate). • Assist students in implementing the algorithm using the provided library or framework.

**3. Training and Testing the Model:**

• Run the AIS algorithm on the training data for each group.

• Explain the selection, mutation, and cloning processes within the algorithm. • Observe how the "antibodies" (damage detectors) evolve over generations to better recognize the  assigned damage pattern.

• Once training is complete, test the model with unseen data containing both healthy and damaged  structures.

• Evaluate the performance of the AIS model based on its accuracy in detecting the assigned damage  type.

**4. Analysis and Discussion:**

• Each group presents their results and discusses the performance of their AIS model. • Compare the effectiveness of different damage detection algorithms or parameter settings across  groups.

• Discuss the advantages and limitations of using AIS for damage detection.

• Explore potential applications of AIS in real-world engineering scenarios.

**5. Diagram:**

A simplified diagram illustrating the key steps of the lab experiment:

| Structure Data | (Healthy & Damaged)

+--------------------+

|

v

+--------------------+

| Feature Extraction |

+--------------------+

|

v

+--------------------+

| Data Encoding | (Binary Strings)

+--------------------+

**]**

|

v

+--------------------+

| AIS Algorithm | (Clonal Selection)

+--------------------+

|

v

+--------------------+

| Trained Detectors | (Damage Patterns)

+--------------------+

|

v

+--------------------+

| Test Data | (Unseen Structures)

+--------------------+

|

v

+--------------------+

| Damage Detection | (Accuracy Evaluation)

+--------------------+

|

v

+--------------------+

| Results & Discussion| (Performance Analysis)

+--------------------+

**Note:** This is a general outline and can be adapted based on the specific AIS algorithm, chosen  damage types, and available resources.

This code represents a process flow for a damage detection system using an AIS algorithm.

1. Structure Data: Represents the input data for the system, which includes information about  healthy and damaged structures.

2. Feature Extraction: Extracts relevant features from the input data to be used in the detection  process.

3. Data Encoding: Converts the extracted features into binary strings for processing by the AIS  algorithm.

4. AIS Algorithm: Utilizes a Clonal Selection algorithm to detect damage patterns in the  encoded data.

5. Trained Detectors: Represents the detectors that have been trained to recognize specific  damage patterns.

6. Test Data: Contains unseen structures that will be used to test the performance of the  detection system.

7. Damage Detection: Evaluates the accuracy of the detection system in identifying damage in  the test data.

8. Results & Discussion: Analyzes the performance of the system and discusses the results  obtained from the damage detection process.

# Code:

import numpy as np

# Define AIRS algorithm

class AIRS:

    def \_\_init\_\_(self, num\_detectors=10, hypermutation\_rate=0.1):

        self.num\_detectors = num\_detectors

        self.hypermutation\_rate = hypermutation\_rate

    def train(self, X, y):

        # Initialize detectors using k-means clustering or other techniques

        # For simplicity, initializing randomly in this example

        self.detectors = X[np.random.choice(len(X), self.num\_detectors, replace=False)]

    def predict(self, X):

        predictions = []

        for sample in X:

            distances = np.linalg.norm(self.detectors - sample, axis=1)

            prediction = int(np.argmin(distances))

            predictions.append(prediction)

        return 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)

    return data, labels

# Split data into training and testing sets

def split\_data(data, labels, 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:]

    return train\_data, test\_data, train\_labels, test\_labels

# Evaluate accuracy

def evaluate\_accuracy(predictions, true\_labels):

    accuracy = np.mean(predictions == true\_labels)

    return accuracy

# Main function

def main():

    # Generate dummy data

    data, labels = generate\_dummy\_data()

    # Split data into training and testing sets

    train\_data, test\_data, train\_labels, test\_labels = split\_data(data, labels)

    # 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

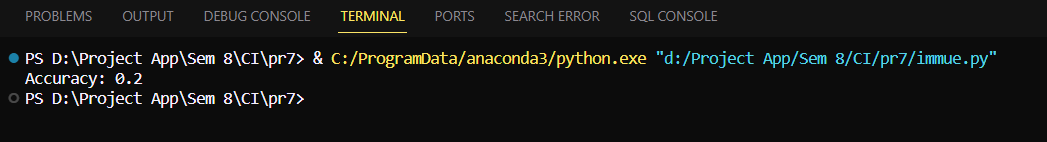
    accuracy = evaluate\_accuracy(predictions, test\_labels)

    print(f"Accuracy: {accuracy}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Output:**

****