Artificial_Immune_Pattern_Recognition

April 2, 2025

[7]:
```
pip install matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: matplotlib in
c:\users\student\anaconda3\lib\site-packages (3.7.1)
Requirement already satisfied: seaborn in c:\users\student\anaconda3\lib\site-
packages (0.12.2)
Requirement already satisfied: scikit-learn in
c:\users\student\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\student\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pandas>=0.25 in
c:\users\student\anaconda3\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: scipy>=1.5.0 in
c:\users\student\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in
c:\users\student\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\student\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\student\anaconda3\lib\site-packages (from pandas>=0.25->seaborn)
(2022.7)
Requirement already satisfied: six>=1.5 in c:\users\student\anaconda3\lib\site-
```

```
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

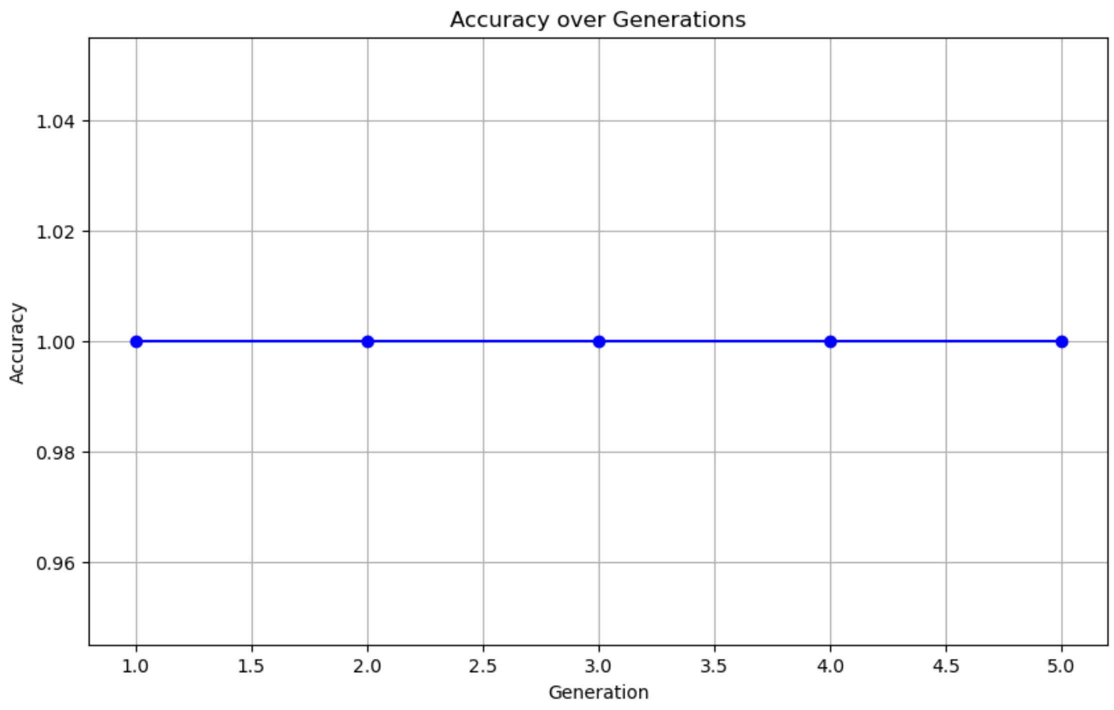
[8]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np
```

[9]:
```python
class ArtificialImmuneSystem:
    def __init__(self, population_size=10, generations=10):
        self.population_size = population_size
        self.generations = generations
        self.population = []  # This will store our "antibodies"
        self.best_antibody = None
        self.accuracies_over_generations = []  # Track accuracies over␣
↪generations

    def initialize_population(self):
        """ Initialize a population of antibodies (random classifiers) """
        for _ in range(self.population_size):
            clf = RandomForestClassifier(n_estimators=10)  # Basic classifier␣
↪as our antibody
            self.population.append(clf)

    def evaluate_population(self, X_train, y_train):
        """ Evaluate the fitness of each antibody (classifier) in the␣
↪population """
        accuracies = []
        for clf in self.population:
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_train)
            accuracy = accuracy_score(y_train, y_pred)
            accuracies.append(accuracy)
        return accuracies

    def select_best_antibodies(self, accuracies):
        """ Select the best antibodies based on accuracy """
        sorted_indices = np.argsort(accuracies)[::-1]
        self.population = [self.population[i] for i in sorted_indices[:self.
↪population_size // 2]]

    def clone_and_mutate(self):
        """ Clone the best antibodies and apply mutations to evolve them """
        clones = self.population.copy()
        for clf in clones:
            if random.random() < 0.5:  # Mutation rate
```

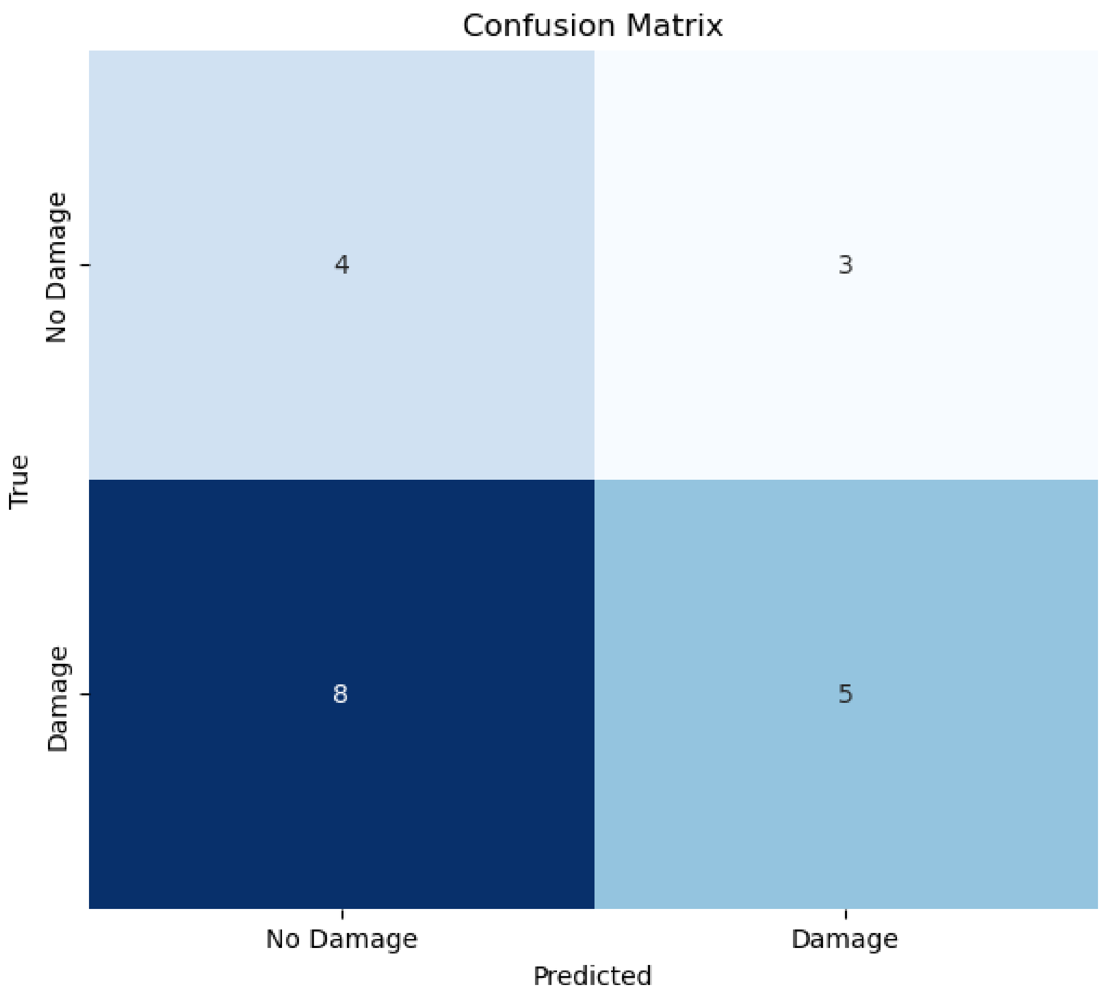
```
                # Example mutation: change classifier parameters or modify the␣
↪dataset
                clf.n_estimators = random.choice([10, 50, 100])
        self.population.extend(clones)

    def train(self, X_train, y_train):
        self.initialize_population()
        for generation in range(self.generations):
            accuracies = self.evaluate_population(X_train, y_train)
            self.select_best_antibodies(accuracies)
            self.clone_and_mutate()
            self.accuracies_over_generations.append(np.max(accuracies))  #␣
↪Track max accuracy of this generation

        # Train the best classifier in the population
        self.best_antibody = self.population[0]

    def predict(self, X_test):
        """ Predict using the best antibody """
        return self.best_antibody.predict(X_test)
```

[10]:

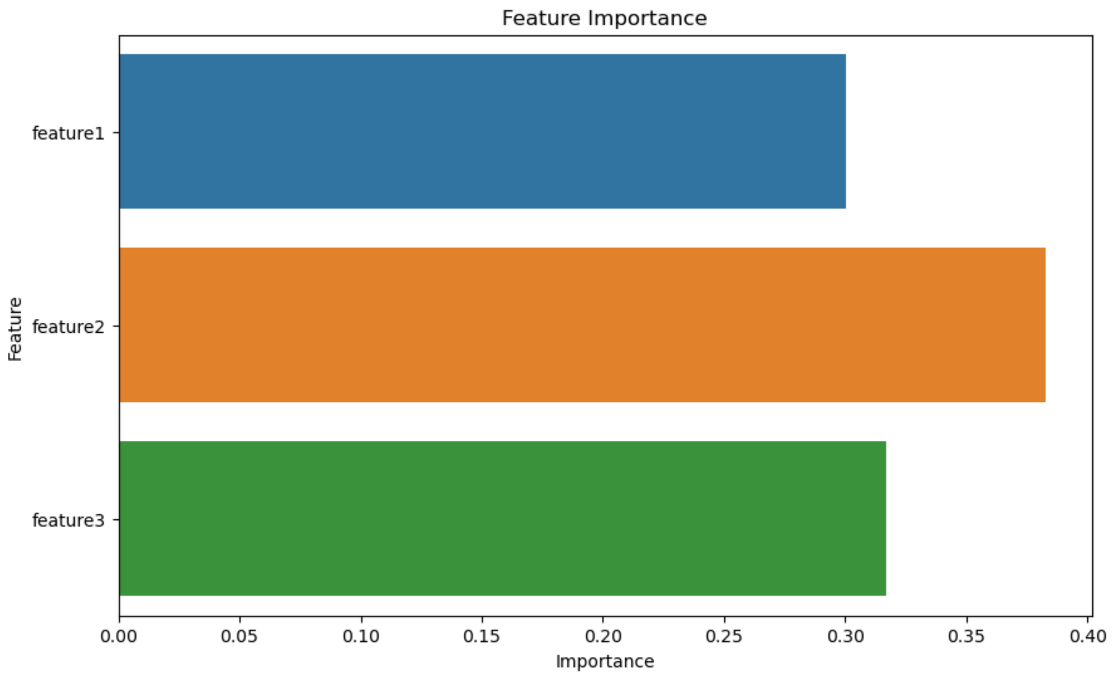
```
# Create and train the AIS model
ais = ArtificialImmuneSystem(population_size=20, generations=5)
ais.train(X_train, y_train)

# Plot the accuracy over generations
plt.figure(figsize=(10, 6))
plt.plot(range(1, ais.generations + 1), ais.accuracies_over_generations,␣
 ↪marker='o', linestyle='-', color='b')
plt.title('Accuracy over Generations')
plt.xlabel('Generation')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```



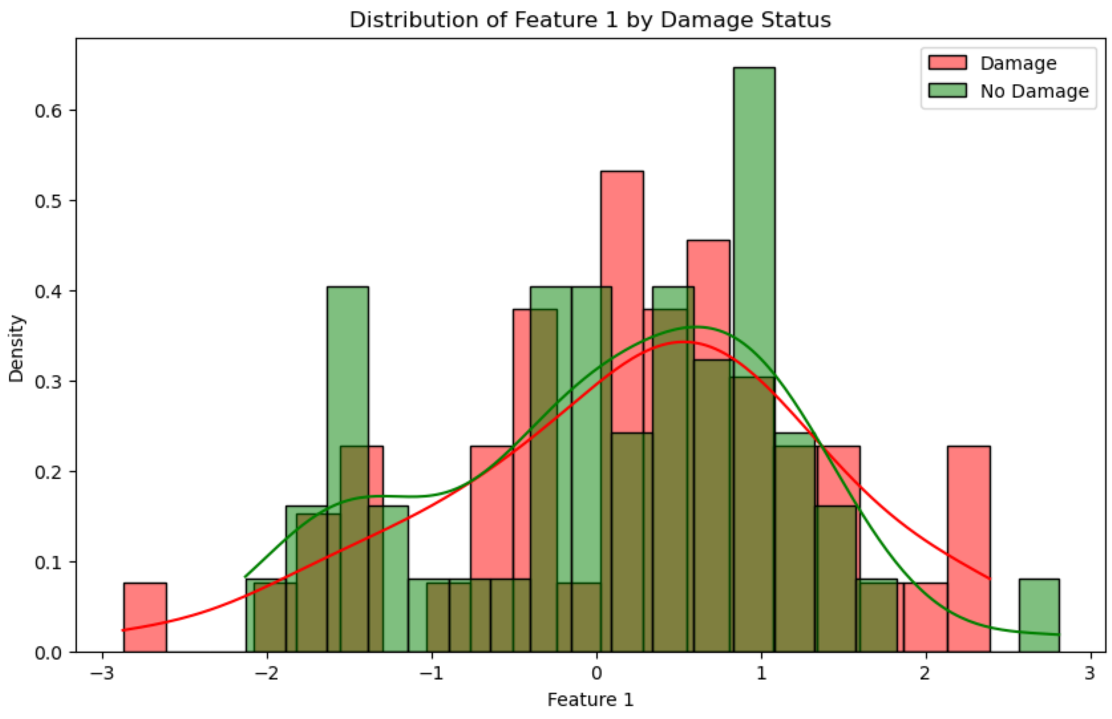
[11]:
```
# Predict on the test data
y_pred = ais.predict(X_test)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(7, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No␣
  ↪Damage", "Damage"], yticklabels=["No Damage", "Damage"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



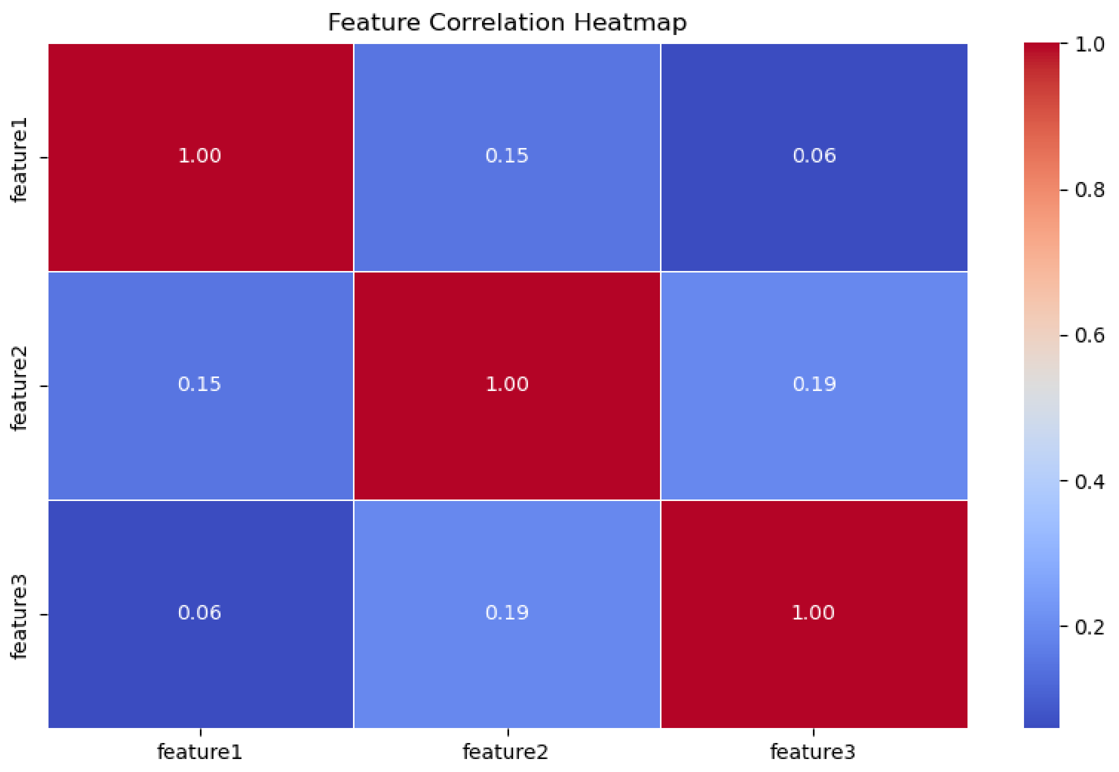
[12]:
```
# Get feature importance from the best antibody (RandomForestClassifier)
importances = ais.best_antibody.feature_importances_
features = X.columns  # Assuming your data is a DataFrame and has column names

# Plot feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x=importances, y=features)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



[13]:
```
# Plot the distribution of a feature (e.g., 'feature1') for both classes␣
↪(Damage/No Damage)
plt.figure(figsize=(10, 6))
sns.histplot(df[df['target'] == 1]['feature1'], color='r', label='Damage',␣
↪kde=True, stat='density', bins=20)
sns.histplot(df[df['target'] == 0]['feature1'], color='g', label='No Damage',␣
↪kde=True, stat='density', bins=20)
plt.title('Distribution of Feature 1 by Damage Status')
plt.xlabel('Feature 1')
plt.ylabel('Density')
plt.legend()
plt.show()
```



[14]:
```python
# Compute correlation matrix
corr_matrix = df.drop(columns='target').corr()

# Plot the correlation heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Feature Correlation Heatmap')
plt.show()
```


Feature Correlation Heatmap
feature1
feature2
feature3
1.00
0.15
0.06
0.15
1.00
0.19
0.06
0.19
1.00
feature1
feature2
feature3
1.0
0.8
0.6
0.4
0.2

```
[ ]:
```