# Documentație pentru proiectul de clasificare a ciupercilor

## Introducere

Această documentație descrie implementarea unui sistem de clasificare a ciupercilor folosind un algoritm evolutiv și o rețea neuronală multi-layer perceptron (MLP). Sistemul permite utilizatorilor să prezice dacă o ciupercă este comestibilă sau otrăvitoare pe baza unor atribute specifice.

## Descrierea problemei considerate

Problema clasificării ciupercilor implică determinarea comestibilității acestora pe baza unui set de atribute, precum forma pălăriei, culoarea, textura, mirosul și altele. Datele utilizate provin dintr-un set de date cunoscut, "Mushroom Data Set", disponibil pe UCI Machine Learning Repository.

## Aspecte teoretice privind algoritmul

### Rețeaua neuronală MLP

Rețeaua MLP utilizată pentru clasificare constă în:

- Un strat de intrare proporțional cu numărul de caracteristici ale datelor.

- Un strat ascuns care reduce dimensiunea la jumătate.

- Un strat de ieșire cu o singură unitate pentru predicții binare (comestibil/otrăvitor).

Funcția de activare folosită este Sigmoid, iar funcția de pierdere este Binary Cross Entropy Loss (BCELoss).

### Algoritmul evolutiv

Algoritmul evolutiv optimizează parametrii rețelelor neuronale prin:

1. Generarea unei populații inițiale.

2. Selecția indivizilor prin turneu.

3. Realizarea operațiilor de crossover și mutație pentru a genera noi indivizi.

4. Calcularea fitness-ului bazat pe eroarea modelului.

## Modalitatea de rezolvare

### Structura proiectului

Proiectul este organizat astfel:

```

project/

data/

attributes.json

size.txt

models/

mushroom\_classifier.pth

scripts/

Perceptron\_Inference.py

EvolutionaryAlgorithm.py

Preceptron\_Training.py

Test\_Accuracy.py

```

### Preprocesarea datelor

Fișierul `DataProcessor.py` efectuează preprocesarea datelor prin:

- Codificarea One-Hot a caracteristicilor categorice.

- Standardizarea valorilor pentru o mai bună convergență a modelului.

### Antrenarea modelului

Modelul este antrenat folosind un algoritm evolutiv care optimizează parametrii rețelelor neuronale pentru a minimiza pierderea.

### Interfața utilizatorului

Interfața grafică construită cu Tkinter permite utilizatorilor să selecteze atributele ciupercii și să obțină predicții.

## Listarea părților semnificative din cod

### Extragerea datelor

```python

# Fetch dataset

mushroom = fetch\_ucirepo(id=73)

x = mushroom.data.features

y = mushroom.data.targets.replace({'e': 0, 'p': 1})

```

### Definirea rețelei MLP

```python

class MLP(nn.Module):

def \_\_init\_\_(self, input\_size):

super(MLP, self).\_\_init\_\_()

self.input\_layer = nn.Linear(input\_size, input\_size // 2)

self.hidden\_layer = nn.Linear(input\_size // 2, 1)

self.sigmoid = nn.Sigmoid()

def forward(self, x):

x = torch.sigmoid(self.input\_layer(x))

x = self.sigmoid(self.hidden\_layer(x))

return x

```

### Algoritmul evolutiv

```python

class EvolutionaryAlgorithm:

def solve(self, problem, population\_size, max\_generations, crossover\_rate, mutation\_rate):

# Initialize population

population = [problem.make\_chromosome() for \_ in range(population\_size)]

for gen in range(max\_generations):

new\_population = [Selection.get\_best(population)] # Elitism

for \_ in range(1, population\_size):

father = Selection.tournament(population)

mother = Selection.tournament(population)

child = Crossover.arithmetic(mother, father, crossover\_rate)

Mutation.reset(child, mutation\_rate)

problem.compute\_fitness(child)

new\_population.append(child)

population = new\_population

return Selection.get\_best(population)

```

## Rezultatele obținute

### Capturi de ecran

A screenshot of a computer

Description automatically generated

### Comentarii

Modelul a obținut o acuratețe de peste 90% pe setul de testare. Algoritmul evolutiv a redus semnificativ timpul de optimizare comparativ cu metodele tradiționale de antrenare.

## Concluzii

Proiectul demonstrează aplicabilitatea algoritmilor evolutivi în optimizarea rețelelor neuronale pentru sarcini de clasificare. Interfața utilizator permite o utilizare ușoară, iar rezultatele confirmă eficiența soluției propuse.

## Bibliografie

1. UCI Machine Learning Repository - Mushroom Data Set

2. Florin Leon - Algoritmi Evolutivi, note de curs

3. Documentația PyTorch

##Lista membrii

1. Antoci Alexandru-Andrei

-Test\_Accuracy.py

-Perceptron\_Interface.py

-Identificat metode de encodat setul de date

-Gasit metoda de identificare a accuracy-ului

-Documentatie parte generala si analiza cod

1. Dimitrie Leustean

-Perceptron\_Training.py

-Configurare Hiperparametrii(Population Size, Epochs, Perceptron Architecture/Nr. Layers/Nr. Neurons)

-Documentatie pentru Perceptron

1. Iovite Tudor

-EvolutionaryAlgorithm.py

-Asistenta la clasa MLPEvolutionProblem

-Documentatie pentru Algoritm Evolutiv