import numpy as np

# Oyuncu sayısı

num\_players = 2

# Oyuncuların stratejik seçenekleri

actions = ['A', 'B', 'C']

# Oyuncuların stratejilerini rastgele başlatma

player\_strategies = []

for \_ in range(num\_players):

player\_strategies.append(np.random.choice(actions))

# Oyuncuların ödeme matrisini oluşturma

payoff\_matrix = np.array([[3, 2, 1], [1, 3, 2], [2, 1, 3]])

# Makine öğrenme algoritması (Q-Learning)

Q = np.zeros((len(actions), len(actions)))

def calculate\_payoff(strategies):

# Oyuncuların stratejilerini kullanarak ödemeleri hesapla

payoffs = []

for player in range(num\_players):

strategy\_indices = [actions.index(strategies[p])

for p in range(num\_players) if p != player]

payoff = np.dot(payoff\_matrix[strategy\_indices]

[:, player], [actions.index(strategies[player])])

payoffs.append(payoff)

return payoffs

def choose\_action(strategy):

# Epsilon-Greedy yöntemiyle bir aksiyon seçme

if np.random.uniform(0, 1) < epsilon:

return np.random.choice(actions)

else:

return actions[np.argmax(Q[strategy])]

# Parametreler

epsilon = 0.2 # Keşif olasılığı

alpha = 0.1 # Öğrenme hızı

gamma = 0.9 # İndirim faktörü

# Oyunu oynatma

for iteration in range(10):

# Oyuncuların stratejilerini yazdırma

print("Iteration", iteration + 1)

for player in range(num\_players):

print("Player", player + 1, "Strategy:", player\_strategies[player])

# Oyuncuların ödemelerini hesaplama

payoffs = calculate\_payoff(player\_strategies)

for player in range(num\_players):

print("Player", player + 1, "Payoff:", payoffs[player])

# Oyuncuların stratejilerini güncelleme (Q-Learning)

for player in range(num\_players):

current\_strategy = actions.index(player\_strategies[player])

action = choose\_action(current\_strategy)

action\_index = actions.index(action)

Q[current\_strategy][action\_index] = (1 - alpha) \* Q[current\_strategy][action\_index] + alpha \* (

payoff\_matrix[current\_strategy][action\_index] + gamma \* np.max(Q[action\_index]))

player\_strategies[player] = action