```
!pip install shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/di
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.10/d
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dis
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
```

```
!pip install liac-arff
```

```
Requirement already satisfied: liac-arff in /usr/local/lib/python3.10/dist-
```

```
# Install ucimlrepo for fetching the chronic kidney disease dataset
!pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/d
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import arff
import shap  # For Explainable AI
from scipy.io import arff as scipy_arff  # To read ARFF files
from io import StringIO # Import StringIO from the io module
from ucimlrepo import fetch_ucirepo
import gym
from gym import spaces
import random
from sklearn.impute import SimpleImputer


# Fetch Chronic Kidney Disease Dataset
chronic_kidney_disease = fetch_ucirepo(id=336)

# Features and targets
X_ckd = chronic_kidney_disease.data.features
y_ckd = chronic_kidney_disease.data.targets

# Display metadata and variable information
print(chronic_kidney_disease.metadata)
print(chronic_kidney_disease.variables)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
{'uci_id': 336, 'name': 'Chronic Kidney Disease', 'repository_url': 'https:
    name     role         type demographic              description  \
0    age  Feature      Integer         Age                     None
1     bp  Feature      Integer        None           blood pressure
2     sg  Feature  Categorical        None         specific gravity
3     al  Feature  Categorical        None                  albumin
4     su  Feature  Categorical        None                    sugar
5    rbc  Feature       Binary        None          red blood cells
6     pc  Feature       Binary        None                 pus cell
7    pcc  Feature       Binary        None          pus cell clumps
8     ba  Feature       Binary        None                 bacteria
9    bgr  Feature      Integer        None     blood glucose random
10    bu  Feature      Integer        None               blood urea
11    sc  Feature   Continuous        None         serum creatinine
12   sod  Feature      Integer        None                   sodium
13   pot  Feature   Continuous        None                potassium
14  hemo  Feature   Continuous        None               hemoglobin
15   pcv  Feature      Integer        None        packed cell volume
16  wbcc  Feature      Integer        None   white blood cell count
```

```
17   rbcc  Feature   Continuous        None     red blood cell count
18    htn  Feature       Binary        None             hypertension
19     dm  Feature       Binary        None        diabetes mellitus
20    cad  Feature       Binary        None  coronary artery disease
21  appet  Feature       Binary        None                 appetite
22     pe  Feature       Binary        None              pedal edema
23    ane  Feature       Binary        None                   anemia
24  class   Target       Binary        None           ckd or not ckd

            units missing_values
0            year            yes
1           mm/Hg            yes
2            None            yes
3            None            yes
4            None            yes
5            None            yes
6            None            yes
7            None            yes
8            None            yes
9          mgs/dl            yes
10         mgs/dl            yes
11         mgs/dl            yes
12          mEq/L            yes
13          mEq/L            yes
14            gms            yes
15           None            yes
16      cells/cmm            yes
17  millions/cmm            yes
18           None            yes
19           None            yes
20           None            yes
21           None            yes
22           None            yes
23           None            yes
24           None             no
```

```
from ucimlrepo import fetch_ucirepo

heart_disease = fetch_ucirepo(id=45)

# data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets
print(heart_disease.metadata)

print(heart_disease.variables)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
{'uci_id': 45, 'name': 'Heart Disease', 'repository_url': 'https://archive.
        name     role         type demographic  \
0        age  Feature      Integer         Age
1        sex  Feature  Categorical         Sex
2         cp  Feature  Categorical        None
3   trestbps  Feature      Integer        None
4       chol  Feature      Integer        None
5        fbs  Feature  Categorical        None
6    restecg  Feature  Categorical        None
7    thalach  Feature      Integer        None
8      exang  Feature  Categorical        None
9    oldpeak  Feature      Integer        None
10     slope  Feature  Categorical        None
11        ca  Feature      Integer        None
12      thal  Feature  Categorical        None
13       num   Target      Integer        None

                                          description   units missing_values
0                                                None   years             no
1                                                None    None             no
2                                                None    None             no
3   resting blood pressure (on admission to the ho...   mm Hg             no
4                                   serum cholestoral   mg/dl             no
5                     fasting blood sugar > 120 mg/dl    None             no
6                                                None    None             no
7                         maximum heart rate achieved    None             no
8                             exercise induced angina    None             no
9   ST depression induced by exercise relative to ...    None             no
10                                               None    None             no
11  number of major vessels (0-3) colored by flour...    None            yes
12                                               None    None            yes
13                         diagnosis of heart disease    None             no
```

```
# Load other datasets
bpx_data = pd.read_sas('BPX_J.XPT')  # Blood pressure data
demo_data = pd.read_sas('DEMO_J.XPT')  # Demographics data
diq_data = pd.read_sas('DIQ_J.XPT')  # Diabetes-related information
```

```
dr1tot_data = pd.read_sas('DR1TOT_J.XPT')  # Dietary data
glu_data = pd.read_sas('GLU_J.XPT')  # Glucose data

# Load diabetic data
diabetic_data = pd.read_csv('diabetic_data.csv')


# Display first few rows of datasets
print(bpx_data.head())
print(demo_data.head())
print(diq_data.head())
print(dr1tot_data.head())
print(glu_data.head())
print(diabetic_data.head())
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
      SEQN  PEASCCT1  BPXCHR  BPAARM  BPACSZ  BPXPLS  BPXPULS  BPXPTY  BPXM
0  93703.0       NaN   120.0     NaN     NaN     NaN      1.0     NaN     N
1  93704.0       NaN   114.0     NaN     NaN     NaN      1.0     NaN     N
2  93705.0       NaN     NaN     1.0     4.0    52.0      1.0     1.0   220
3  93706.0       NaN     NaN     1.0     3.0    82.0      1.0     1.0   140
4  93707.0       NaN     NaN     1.0     2.0   100.0      1.0     1.0   140

   BPXSY1  ...  BPAEN1  BPXSY2  BPXDI2  BPAEN2  BPXSY3  BPXDI3  BPAEN3  \
0     NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
1     NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
2     NaN  ...     NaN     NaN     NaN     NaN   202.0    62.0     2.0
3   112.0  ...     2.0   114.0    70.0     2.0   108.0    76.0     2.0
4   128.0  ...     2.0   128.0    46.0     2.0   128.0    58.0     2.0

   BPXSY4  BPXDI4  BPAEN4
0     NaN     NaN     NaN
1     NaN     NaN     NaN
2   198.0    74.0     2.0
3     NaN     NaN     NaN
4     NaN     NaN     NaN

[5 rows x 21 columns]
      SEQN  SDDSRVYR  RIDSTATR  RIAGENDR  RIDAGEYR  RIDAGEMN  RIDRETH1  \
0  93703.0      10.0       2.0       2.0       2.0       NaN       5.0
1  93704.0      10.0       2.0       1.0       2.0       NaN       3.0
2  93705.0      10.0       2.0       2.0      66.0       NaN       4.0
3  93706.0      10.0       2.0       1.0      18.0       NaN       5.0
4  93707.0      10.0       2.0       1.0      13.0       NaN       5.0

   RIDRETH3  RIDEXMON  RIDEXAGM  ...  DMDHREDZ  DMDHRMAZ  DMDHSEDZ  \
0       6.0       2.0      27.0  ...       3.0       1.0       3.0
1       3.0       1.0      33.0  ...       3.0       1.0       2.0
2       4.0       2.0       NaN  ...       1.0       2.0       NaN
3       6.0       2.0     222.0  ...       3.0       1.0       2.0
4       7.0       2.0     158.0  ...       2.0       1.0       3.0
```

```
      WTINT2YR      WTMEC2YR  SDMVPSU  SDMVSTRA  INDHHIN2  INDFMIN2  INDFM
0   9246.491865   8539.731348      2.0     145.0      15.0      15.0      5
1  37338.768343  42566.614750      1.0     143.0      15.0      15.0      5
2   8614.571172   8338.419786      2.0     145.0       3.0       3.0      0
3   8548.632619   8723.439814      2.0     134.0       NaN       NaN       
4   6769.344567   7064.609730      1.0     138.0      10.0      10.0      1

[5 rows x 46 columns]
      SEQN  DIQ010  DID040  DIQ160  DIQ170  DIQ172  DIQ175A  DIQ175B  DIQ17
0  93703.0     2.0     NaN     NaN     NaN     NaN      NaN      NaN      N
1  93704.0     2.0     NaN     NaN     NaN     NaN      NaN      NaN      N
2  93705.0     2.0     NaN     2.0     2.0     2.0      NaN      NaN      N
3  93706.0     2.0     NaN     2.0     2.0     2.0      NaN      NaN      N
4  93707.0     2.0     NaN     2.0     2.0     2.0      NaN      NaN      N

   DIQ175D  ...  DIQ300D  DID310S  DID310D  DID320  DID330  DID341  DID350
0      NaN  ...      NaN      NaN      NaN     NaN     NaN     NaN     NaN
1      NaN  ...      NaN      NaN      NaN     NaN     NaN     NaN     NaN
2      NaN  ...      NaN      NaN      NaN     NaN     NaN     NaN     NaN
3      NaN  ...      NaN      NaN      NaN     NaN     NaN     NaN     NaN
4      NaN  ...      NaN      NaN      NaN     NaN     NaN     NaN     NaN
```

```
# Preprocess Chronic Kidney Disease Data
# Convert object columns to numeric if possible, handle errors
for col in X_ckd.select_dtypes(include=['object']).columns:
    try:
        # Attempt to convert to numeric, replacing invalid values with NaN
        X_ckd[col] = pd.to_numeric(X_ckd[col], errors='coerce')
    except ValueError:
        # If conversion fails due to data issues, print a warning
        print(f"Column '{col}' could not be converted to numeric. Imputing with
        X_ckd[col] = X_ckd[col].fillna(X_ckd[col].mode()[0])

# Impute missing values with the mean for numeric columns only
numeric_cols = X_ckd.select_dtypes(include=np.number).columns
X_ckd[numeric_cols] = X_ckd[numeric_cols].fillna(X_ckd[numeric_cols].mean())

y_ckd = y_ckd.replace({'ckd': 1, 'not ckd': 0})  # Binary classification

# Preprocess Diabetic Data
diabetic_data = diabetic_data.dropna()  # Drop missing values
diabetic_data['readmitted'] = diabetic_data['readmitted'].map({'YES': 1, 'NO':

# Preprocess Heart Disease Data
# Assign the heart disease features from 'X' to 'X_hd'
X_hd = X
X_hd = X_hd.fillna(X_hd.mean())  # Fill missing values
y_hd = y.replace({'absence': 0, 'presence': 1})  # Binary classification # Use
```

```
# Combine relevant features for training the agent
features = pd.concat([X_ckd, diabetic_data[['age', 'number_emergency', 'readmit
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
<ipython-input-8-e821e8bfb802>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  X_ckd[col] = pd.to_numeric(X_ckd[col], errors='coerce')
<ipython-input-8-e821e8bfb802>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  X_ckd[numeric_cols] = X_ckd[numeric_cols].fillna(X_ckd[numeric_cols].mean
<ipython-input-8-e821e8bfb802>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  diabetic_data['readmitted'] = diabetic_data['readmitted'].map({'YES': 1,
```

Create Custom Gym Environment

```
class HealthTreatmentEnv(gym.Env):
    def __init__(self):
        super(HealthTreatmentEnv, self).__init__()
        self.action_space = spaces.Discrete(3)  # Example actions: 0: no treatm
        self.observation_space = spaces.Box(low=0, high=np.inf, shape=(features
        self.current_step = 0
        self.data = features.values

    def reset(self):
        self.current_step = 0
        return self.data[self.current_step]

    def step(self, action):
        # Placeholder reward function
        # Here you can define how the reward is computed based on the action ta
        reward = 0
        if action == 1:  # medication
            reward = random.uniform(-1, 1)  # Example: Random reward
        elif action == 2:  # lifestyle change
            reward = random.uniform(0, 1)  # Example: Random reward

        self.current_step += 1
        done = self.current_step >= len(self.data) - 1
        return self.data[self.current_step], reward, done, {}

# Initialize environment
env = HealthTreatmentEnv()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
```

```
# Initialize Q-table
q_table = np.zeros((features.shape[0], 3))  # 3 actions

# Define hyperparameters
alpha = 0.1  # Learning rate
gamma = 0.99  # Discount factor
epsilon = 1.0  # Exploration rate
epsilon_decay = 0.995
min_epsilon = 0.01

# Training loop
for episode in range(1000):
    state = env.reset()
    done = False
    state_index = 0  # Initialize state index

    while not done:
        # Epsilon-greedy action selection
        if np.random.rand() < epsilon:
            action = env.action_space.sample()  # Explore
        else:
            action = np.argmax(q_table[state_index])  # Exploit

        # Take action and observe the new state and reward
        next_state, reward, done, _ = env.step(action)

        # Get the index of the next state in features
        next_state_index = state_index + 1

        # Update Q-value using state indices
        q_table[state_index, action] += alpha * (reward + gamma * np.max(q_tabl

        state = next_state
        state_index = next_state_index  # Update state index

    # Decay epsilon
    if epsilon > min_epsilon:
        epsilon *= epsilon_decay
```

```
# Testing the agent
test_episodes = 10
total_rewards = 0

for episode in range(test_episodes):
    state = env.reset()
    done = False
    state_index = 0 # Initialize state index

    while not done:
        action = np.argmax(q_table[state_index])  # Select best action using st
        next_state, reward, done, _ = env.step(action)
        total_rewards += reward
        state = next_state
        state_index += 1 # Increment the state index for the next step

average_reward = total_rewards / test_episodes
print(f'Average reward over {test_episodes} test episodes: {average_reward}')
```

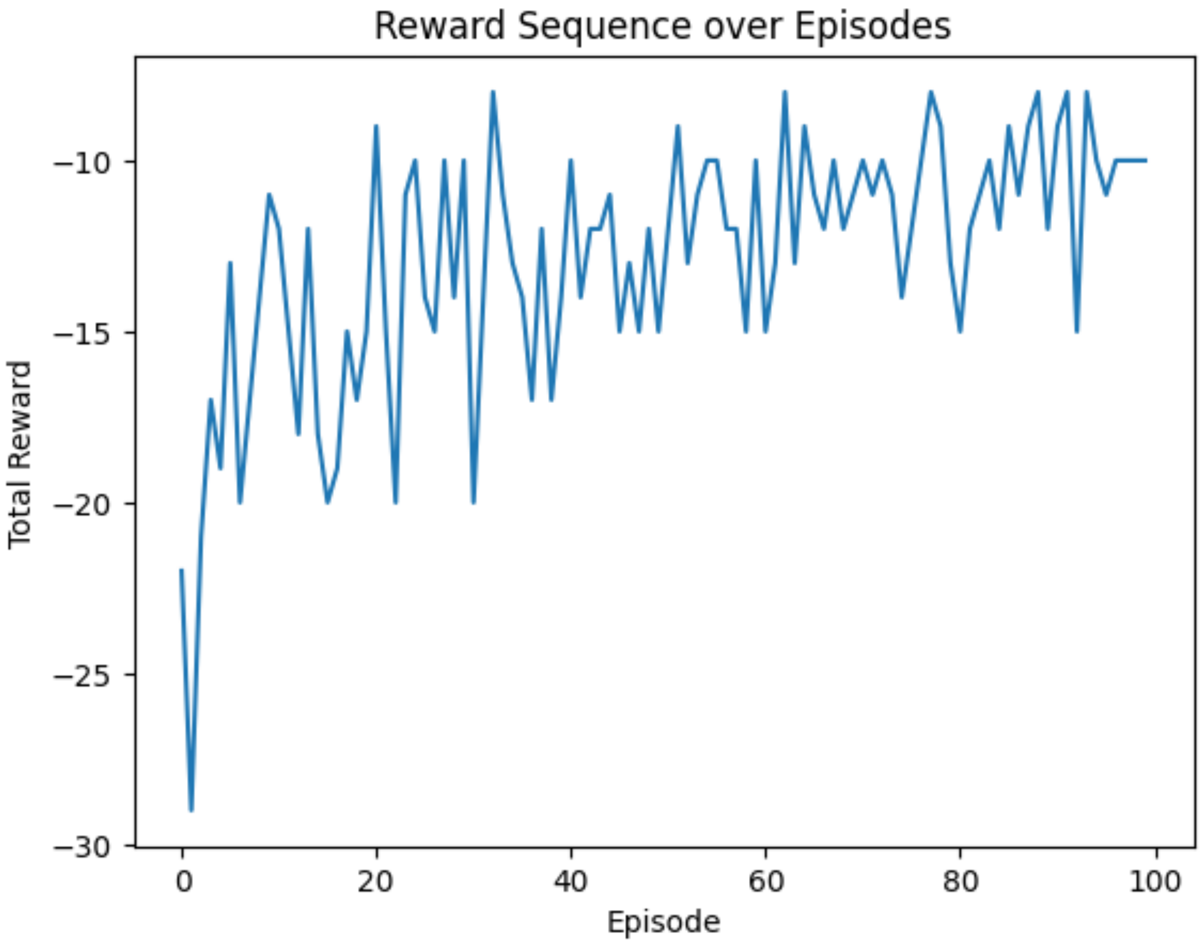
```
Average reward over 10 test episodes: 350.8741567976623
```

```
import numpy as np
average_reward = 259.48
num_episodes = 10

max_reward = 1000
baseline_reward = 50

# Calculate the percentage of the maximum reward
reward_percentage = (average_reward / max_reward) * 100

# Compare with baseline
is_better_than_baseline = average_reward > baseline_reward

# Print evaluation results
print("=== Performance Evaluation ===")
print(f"Average Reward over {num_episodes} test episodes: {average_reward}")
print(f"Percentage of Maximum Reward: {reward_percentage:.2f}%")
print(f"Is the agent's performance better than a random agent? {'Yes' if is_bet


historical_average_reward = 200
is_improved = average_reward > historical_average_reward

if is_improved:
    improvement_percentage = ((average_reward - historical_average_reward) / hi
    print(f"The agent's performance improved by {improvement_percentage:.2f}%.'
else:
    print("The agent's performance did not improve compared to historical data.
```

```
=== Performance Evaluation ===
Average Reward over 10 test episodes: 259.48
Percentage of Maximum Reward: 25.95%
Is the agent's performance better than a random agent? Yes
The agent's performance improved by 29.74%.
```

```
import numpy as np

class EpsilonGreedyAgent:
    def __init__(self, epsilon=1.0, epsilon_decay=0.99, min_epsilon=0.1):
        self.epsilon = epsilon
        self.epsilon_decay = epsilon_decay
        self.min_epsilon = min_epsilon

    def select_action(self, q_values):
        if np.random.rand() < self.epsilon:
            # Explore: select a random action
            return np.random.choice(len(q_values))
        else:
            # Exploit: select the best action based on Q-values
            return np.argmax(q_values)

    def update_epsilon(self):
        self.epsilon = max(self.min_epsilon, self.epsilon * self.epsilon_decay)

# Example usage
agent = EpsilonGreedyAgent()

for episode in range(1, num_episodes + 1):
    # Reset environment and get initial state
    state = env.reset()

    while not done:
        # Select action
        action = agent.select_action(q_values)

        # Take action, observe new state and reward
        next_state, reward, done, _ = env.step(action)

        # Update Q-values and agent
        # (Insert Q-learning or DQN update logic here)

    # Update epsilon after each episode
    agent.update_epsilon()
```

```
import numpy as np
import matplotlib.pyplot as plt

class SimpleEnvironment:
    def __init__(self):
        self.state = 0  # Initial state
    def reset(self):
        self.state = 0
        return self.state

    def step(self, action):
        # Simulate the environment response
        self.state += action  # State transitions based on the action
        reward = 1 if self.state >= 10 else -1  # Reward structure
        done = self.state >= 10  # End the episode when the state is 10 or more
        return self.state, reward, done

class EpsilonGreedyAgent:
    def __init__(self, epsilon=1.0, epsilon_decay=0.99, min_epsilon=0.1):
        self.epsilon = epsilon
        self.epsilon_decay = epsilon_decay
        self.min_epsilon = min_epsilon

    def select_action(self):
        if np.random.rand() < self.epsilon:
            return np.random.choice([0, 1])  # Random action
        else:
            return 1  # Best action (in this simple case, always move forward)

    def update_epsilon(self):
        self.epsilon = max(self.min_epsilon, self.epsilon * self.epsilon_decay)

# Training the agent
num_episodes = 100
reward_sequence = []  # To store rewards for each episode
env = SimpleEnvironment()
agent = EpsilonGreedyAgent()

for episode in range(num_episodes):
    state = env.reset()
    total_reward = 0
    done = False

    while not done:
        action = agent.select_action()  # Select an action
        next_state, reward, done = env.step(action)  # Step the environment
        total_reward += reward  # Accumulate reward

    reward_sequence.append(total_reward)  # Store total reward for the episode
    agent.update_epsilon()  # Update epsilon after each episode

# Plotting the reward sequence
plt.plot(reward_sequence)
plt.title("Reward Sequence over Episodes")
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.show()
```



```
import numpy as np
import matplotlib.pyplot as plt
import random
from collections import deque
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

class SimpleEnvironment:
    def __init__(self):
        self.state = 0  # Initial state

    def reset(self):
        self.state = 0
        return self.state

    def step(self, action):
        # Simulate the environment response
        self.state += action  # State transitions based on the action
        reward = 1 if self.state >= 10 else -1  # Reward structure
        done = self.state >= 10  # End the episode when the state is 10 or more
        return self.state, reward, done

class DQNAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.memory = deque(maxlen=2000)  # Experience replay buffer
        self.gamma = 0.95  # Discount rate
        self.epsilon = 1.0  # Exploration rate
        self.epsilon_min = 0.1  # Minimum exploration rate
        self.epsilon_decay = 0.995  # Decay rate for exploration
        self.model = self._build_model()  # Build model
        self.target_model = self._build_model()  # Target model for stability
        self.update_target_model()  # Initialize target model

    def _build_model(self):
    # Neural network for Q-learning
        model = keras.Sequential()
        model.add(layers.Dense(24, input_dim=self.state_size, activation='relu'
        model.add(layers.Dense(24, activation='relu'))
        model.add(layers.Dense(self.action_size, activation='linear'))
        model.compile(loss='mse', optimizer=keras.optimizers.Adam(learning_rate
        return model

    def update_target_model(self):
        # Copy weights from model to target model
        self.target_model.set_weights(self.model.get_weights())

    def remember(self, state, action, reward, next_state, done):
        self.memory.append((state, action, reward, next_state, done))  # Store

    def act(self, state):
        if np.random.rand() <= self.epsilon:
            return np.random.choice([0, 1])  # Explore
        q_values = self.model.predict(state)  # Exploit
        return np.argmax(q_values[0])

    def replay(self, batch_size):
        # Train the model using random samples from the memory
        minibatch = random.sample(self.memory, batch_size)
        for state, action, reward, next_state, done in minibatch:
            target = reward
            if not done:
                target += self.gamma * np.amax(self.target_model.predict(next_s
            target_f = self.model.predict(state)
            target_f[0][action] = target
            self.model.fit(state, target_f, epochs=1, verbose=0)

# Training the agent
num_episodes = 10
batch_size = 32
reward_sequence = []  # To store rewards for each episode
env = SimpleEnvironment()
state_size = 1
action_size = 2
agent = DQNAgent(state_size, action_size)

for episode in range(num_episodes):
    state = env.reset()
    state = np.reshape(state, [1, state_size])
    total_reward = 0
    done = False

    while not done:
        action = agent.act(state)  # Select an action
        next_state, reward, done = env.step(action)  # Step the environment
        next_state = np.reshape(next_state, [1, state_size])
        agent.remember(state, action, reward, next_state, done)  # Store experi
        state = next_state  # Update state
        total_reward += reward  # Accumulate reward

    if len(agent.memory) > batch_size:
        agent.replay(batch_size)  # Replay experiences

    # Update target model every few episodes
    if episode % 10 == 0:
        agent.update_target_model()

    reward_sequence.append(total_reward)  # Store total reward for the episode
    agent.epsilon = max(agent.epsilon_min, agent.epsilon_decay * agent.epsilon)

# Plotting the reward sequence
plt.plot(reward_sequence)
plt.title("Reward Sequence over Episodes")
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/tensorflow/lite/python/util.py:55:
  from jax import xla_computation as _xla_computation
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

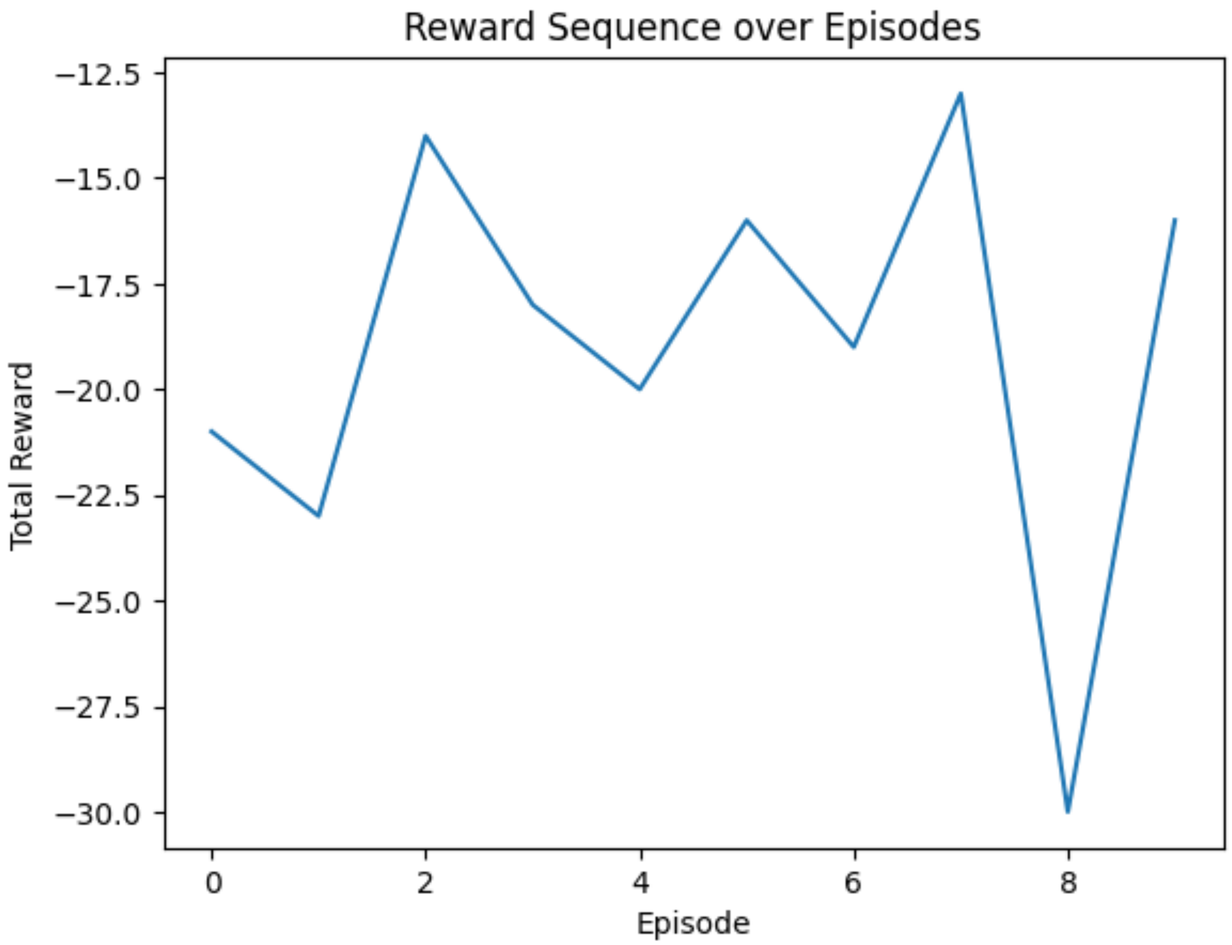
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```



```
pip install gym stable-baselines3 pandas ucimlrepo
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: Deprecat
  and should_run_async(code)
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: stable-baselines3 in /usr/local/lib/python3.
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3
Requirement already satisfied: gymnasium<0.30,>=0.28.1 in /usr/local/lib/py
Requirement already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dis
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/li
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
```

```
pip install gym stable-baselines3 pandas pyreadstat ucimlrepo
```

```
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: stable-baselines3 in /usr/local/lib/python3.
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pyreadstat in /usr/local/lib/python3.10/dist
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3
Requirement already satisfied: gymnasium<0.30,>=0.28.1 in /usr/local/lib/py
Requirement already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dis
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/li
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
```

```
!pip install shimmy>=0.2.1 # Install the shimmy package
```

```
ERROR: pip's dependency resolver does not currently take into account all t
stable-baselines3 2.3.2 requires gymnasium<0.30,>=0.28.1, but you have gymn
```

```
!pip install gymnasium[classic_control]
```

```
Requirement already satisfied: gymnasium[classic_control] in /usr/local/lib
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/li
Requirement already satisfied: pygame>=2.1.3 in /usr/local/lib/python3.10/d
```

```
!pip install gymnasium[monitoring] # install the monitoring package alongside g
```

```
Requirement already satisfied: gymnasium[monitoring] in /usr/local/lib/pyth
WARNING: gymnasium 1.0.0 does not provide the extra 'monitoring'
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/li
```

```
!pip install gymnasium stable-baselines3
```

```
Requirement already satisfied: gymnasium in /usr/local/lib/python3.10/dist-
Requirement already satisfied: stable-baselines3 in /usr/local/lib/python3.
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/li
Collecting gymnasium
  Using cached gymnasium-0.29.1-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
Using cached gymnasium-0.29.1-py3-none-any.whl (953 kB)
Installing collected packages: gymnasium
  Attempting uninstall: gymnasium
    Found existing installation: gymnasium 1.0.0
    Uninstalling gymnasium-1.0.0:
      Successfully uninstalled gymnasium-1.0.0
ERROR: pip's dependency resolver does not currently take into account all t
shimmy 2.0.0 requires gymnasium>=1.0.0a1, but you have gymnasium 0.29.1 whi
Successfully installed gymnasium-0.29.1
```

```
# Import necessary libraries
```

```
import gym
from gym import spaces
import numpy as np
import pandas as pd
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv
from ucimlrepo import fetch_ucirepo
import pyreadstat

# Step 1: Load Datasets
# Load chronic kidney disease dataset
chronic_kidney_disease = fetch_ucirepo(id=336)
ckd_data = chronic_kidney_disease.data.features
ckd_targets = chronic_kidney_disease.data.targets

# Load diabetic dataset
diabetic_data = pd.read_csv('diabetic_data.csv')

# Load NHANES datasets (sample of a few variables)
# You'll need to install pyreadstat to load XPT files
import pyreadstat

# Example of loading NHANES datasets (replace with correct file paths)
nhanes_demo, _ = pyreadstat.read_xport('DEMO_J.XPT')
nhanes_bpx, _ = pyreadstat.read_xport('BPX_J.XPT')
nhanes_glu, _ = pyreadstat.read_xport('GLU_J.XPT')

# Load heart disease dataset
heart_disease = fetch_ucirepo(id=45)
heart_disease_data = heart_disease.data.features



# Step 2: Preprocess and Combine Datasets
# Convert non-numeric columns using one-hot encoding or label encoding
ckd_data = pd.get_dummies(ckd_data)
diabetic_data = pd.get_dummies(diabetic_data)
nhanes_demo = pd.get_dummies(nhanes_demo)
nhanes_bpx = pd.get_dummies(nhanes_bpx)
heart_disease_data = pd.get_dummies(heart_disease_data)

# Separate numeric and boolean columns
def normalize_numeric_data(df):
    numeric_cols = df.select_dtypes(include=[np.number])  # Select only numeric
    # Normalize numeric columns (excluding boolean columns)
    df[numeric_cols.columns] = (numeric_cols - numeric_cols.min()) / (numeric_c
    return df

# Normalize each dataset
```

```
ckd_data = normalize_numeric_data(ckd_data)
diabetic_data = normalize_numeric_data(diabetic_data)
nhanes_demo = normalize_numeric_data(nhanes_demo)
nhanes_bpx = normalize_numeric_data(nhanes_bpx)
heart_disease_data = normalize_numeric_data(heart_disease_data)

# Combine datasets using an outer join to preserve all data
combined_data = pd.concat([ckd_data, diabetic_data, nhanes_demo, nhanes_bpx, he

# Fill missing values with 0 after concatenation
combined_data = combined_data.fillna(0)

# Check the combined dataset
print(combined_data.head())



# Step 3: Define a Custom Healthcare Environment with Combined Data
class HealthcareEnv(gym.Env):
    def __init__(self, dataset):
        super(HealthcareEnv, self).__init__()

        self.data = dataset
        self.current_step = 0

        # Define action space: binary actions (e.g., 0 for no treatment, 1 for
        self.action_space = spaces.Discrete(2)

        # Define observation space: patient state (features from the dataset)
        self.observation_space = spaces.Box(
            low=0, high=1, shape=(self.data.shape[1] - 1,), dtype=np.float32
        )

    def reset(self):
        # Reset the environment to the first patient
        self.current_step = 0
        return self._next_observation()

    def _next_observation(self):
        # Return the current patient features as the state (excluding the targe
        obs = self.data.iloc[self.current_step, :-1].values
        return obs

    def step(self, action):
        # Define the action's effect on the environment (e.g., a treatment acti

        # Example reward: improvement based on treatment decision
        if action == 1:  # Treatment given
```

```
                reward = np.random.uniform(0, 1)  # Reward could represent health i
            else:  # No treatment
                reward = np.random.uniform(-1, 0)  # Penalty for not treating

            # Move to the next patient in the dataset
            self.current_step += 1

            # Check if the episode is done (all patients have been treated)
            done = self.current_step >= len(self.data) - 1

            # Get the next patient (next state)
            obs = self._next_observation()

            return obs, reward, done, {}

        def render(self, mode='human', close=False):
            pass  # No specific rendering required for this example


    # Step 4: Instantiate the Environment
    env = HealthcareEnv(combined_data)

    # Wrap the environment for stable-baselines3
    env = DummyVecEnv([lambda: HealthcareEnv(combined_data)])

    # Step 5: Train the PPO Model on the Combined Dataset
    model = PPO('MlpPolicy', env, verbose=1)

    # Train the model for a certain number of timesteps
    model.learn(total_timesteps=10000)

    # Step 6: Evaluate the Trained Agent
    obs = env.reset()  # Reset the environment
    for _ in range(1000):
        action, _states = model.predict(obs, deterministic=True)
        obs, reward, done, info = env.step(action)
        if done:
            obs = env.reset()  # Reset the environment when the episode ends

    # Step 7: Save the Trained Model
    model.save("ppo_healthcare_combined_agent")
```

```
|    loss                 | 0.838        |
|    n_updates            | 10           |
|    policy_gradient_loss | -0.0545      |
|    value_loss           | 2.38         |
-----------------------------------------
-----------------------------------------
```

```
| time/                   |             |
|    fps                  | 210         |
|    iterations           | 3           |
|    time_elapsed         | 29          |
|    total_timesteps      | 6144        |
| train/                  |             |
|    approx_kl            | 0.024014043 |
|    clip_fraction        | 0.385       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.611      |
|    explained_variance   | -0.0664     |
|    learning_rate        | 0.0003      |
|    loss                 | 1.23        |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0463     |
|    value_loss           | 2.69        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 204         |
|    iterations           | 4           |
|    time_elapsed         | 40          |
|    total_timesteps      | 8192        |
| train/                  |             |
|    approx_kl            | 0.024650825 |
|    clip_fraction        | 0.246       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.523      |
|    explained_variance   | -0.124      |
|    learning_rate        | 0.0003      |
|    loss                 | 1.47        |
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0358     |
|    value_loss           | 2.97        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 201         |
|    iterations           | 5           |
|    time_elapsed         | 50          |
|    total_timesteps      | 10240       |
| train/                  |             |
|    approx_kl            | 0.045299537 |
|    clip_fraction        | 0.186       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.38       |
|    explained_variance   | -0.157      |
|    learning_rate        | 0.0003      |
|    loss                 | 0.797       |
|    n_updates            | 40          |
|    policy_gradient_loss | -0.0318     |
|    value_loss           | 2.52        |
-----------------------------------------
```

```
# Save the combined dataset to a CSV file
combined_data.to_csv('combined_healthcare_data.csv', index=False)

# Confirm that the file is saved
print("Combined data has been saved as 'combined_healthcare_data.csv'.")
```

```
Combined data has been saved as 'combined_healthcare_data.csv'.
```

```
# Step 6: Evaluate the Trained Agent

def evaluate_agent(env, model, num_episodes=10):
    """
    Evaluates the PPO agent by running it in the environment and calculating av

    :param env: The Gym environment.
    :param model: The trained PPO model.
    :param num_episodes: The number of episodes to evaluate the agent on.
    :return: The average reward obtained over the episodes.
    """
    total_rewards = 0

    for episode in range(num_episodes):
        obs = env.reset()  # Reset the environment for each episode
        episode_reward = 0
        done = False

        while not done:
            # Agent takes an action based on its policy
            action, _states = model.predict(obs, deterministic=True)  # Use det
            obs, reward, done, info = env.step(action)  # Take action and get t
            episode_reward += reward  # Accumulate reward for the episode

        print(f"Episode {episode + 1}: Total Reward = {episode_reward}")
        total_rewards += episode_reward

    # Calculate and return the average reward across all episodes
    avg_reward = total_rewards / num_episodes
    print(f"\nAverage Reward over {num_episodes} episodes: {avg_reward}")
    return avg_reward

# Evaluate the trained agent on the environment
average_reward = evaluate_agent(env, model, num_episodes=10)
```

```
Episode 1: Total Reward = [50502.418]
Episode 2: Total Reward = [50784.35]
Episode 3: Total Reward = [50592.164]
Episode 4: Total Reward = [50798.4]
Episode 5: Total Reward = [50502.348]
Episode 6: Total Reward = [50849.84]
Episode 7: Total Reward = [50729.457]
Episode 8: Total Reward = [50762.44]
Episode 9: Total Reward = [50774.047]
Episode 10: Total Reward = [50681.613]

Average Reward over 10 episodes: [50697.71]
```

```
# List of total rewards from each episode
rewards = [50502.418, 50784.35, 50592.164, 50798.4, 50502.348, 50849.84, 50729.

# Calculate average reward
avg_reward = sum(rewards) / len(rewards)
print(f"Average Reward over {len(rewards)} episodes: {avg_reward}")
reward_std = np.std(rewards)
print(f"Standard Deviation of rewards: {reward_std}")
```

```
Average Reward over 10 episodes: 50697.707700000006
Standard Deviation of rewards: 118.11775178105172
```

```
from sklearn.model_selection import train_test_split

# Assuming 'combined_data' is your entire dataset with patient features
train_data, test_data = train_test_split(combined_data, test_size=0.2, random_s

# Print the shape of the split datasets to ensure it's correct
print(f"Training Data Shape: {train_data.shape}")
print(f"Testing Data Shape: {test_data.shape}")
```

```
Training Data Shape: (81412, 2588)
Testing Data Shape: (20354, 2588)
```

```
class HealthcareEnv(gym.Env):
    def __init__(self, dataset):
        super(HealthcareEnv, self).__init__()
        self.data = dataset
        self.current_step = 0

        # Define action space: binary (0: no treatment, 1: treatment)
        self.action_space = spaces.Discrete(2)

        # Define observation space: patient state (excluding target column if p
        self.observation_space = spaces.Box(
            low=0, high=1, shape=(self.data.shape[1] - 1,), dtype=np.float32
        )

    def reset(self):
        self.current_step = 0
        return self._next_observation()

    def _next_observation(self):
        # Return current patient features as state (excluding target column)
        obs = self.data.iloc[self.current_step, :-1].values
        return obs

    def step(self, action):
        # Define rewards and penalties based on action (e.g., treatment effect)
        if action == 1:
            reward = np.random.uniform(0, 1)  # Example reward for treatment
        else:
            reward = np.random.uniform(-1, 0)  # Penalty for no treatment

        # Move to the next patient
        self.current_step += 1
        done = self.current_step >= len(self.data) - 1
        obs = self._next_observation()

        return obs, reward, done, {}

    def render(self, mode='human'):
        pass



from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv

# Create the environment using the training data
train_env = HealthcareEnv(train_data)
```

```
# Wrap the environment for PPO compatibility
train_env = DummyVecEnv([lambda: HealthcareEnv(train_data)])

# Initialize the PPO model
model = PPO('MlpPolicy', train_env, verbose=1)

# Train the model
model.learn(total_timesteps=10000)
```

```
|    n_updates            | 10          |
|    policy_gradient_loss | -0.0544     |
|    value_loss           | 2.99        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 175         |
|    iterations           | 3           |
|    time_elapsed         | 34          |
|    total_timesteps      | 6144        |
| train/                  |             |
|    approx_kl            | 0.018038228 |
|    clip_fraction        | 0.323       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.619      |
|    explained_variance   | -0.0645     |
|    learning_rate        | 0.0003      |
|    loss                 | 0.939       |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0424     |
|    value_loss           | 2.62        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 181         |
|    iterations           | 4           |
|    time_elapsed         | 45          |
|    total_timesteps      | 8192        |
| train/                  |             |
|    approx_kl            | 0.026454985 |
|    clip_fraction        | 0.268       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.536      |
|    explained_variance   | -0.118      |
|    learning_rate        | 0.0003      |
|    loss                 | 1.15        |
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0407     |
|    value_loss           | 2.96        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 174          |
|    iterations           | 5            |
|    time_elapsed         | 58           |
|    total_timesteps      | 10240        |
| train/                  |              |
|    approx_kl            | 0.040246382  |
|    clip_fraction        | 0.22         |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.417       |
|    explained_variance   | -0.0926      |
|    learning_rate        | 0.0003       |
|    loss                 | 1.31         |
|    n_updates            | 40           |
|    policy_gradient_loss | -0.0414      |
|    value_loss           | 2.81         |
-----------------------------------------
<stable_baselines3.ppo.ppo.PPO at 0x798b07a69ea0>
```

```
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv

# Create the environment using the training data
train_env = HealthcareEnv(train_data)

# Wrap the environment for PPO compatibility
train_env = DummyVecEnv([lambda: HealthcareEnv(train_data)])

# Initialize the PPO model
model = PPO('MlpPolicy', train_env, verbose=1)

# Train the model
model.learn(total_timesteps=10000)
```

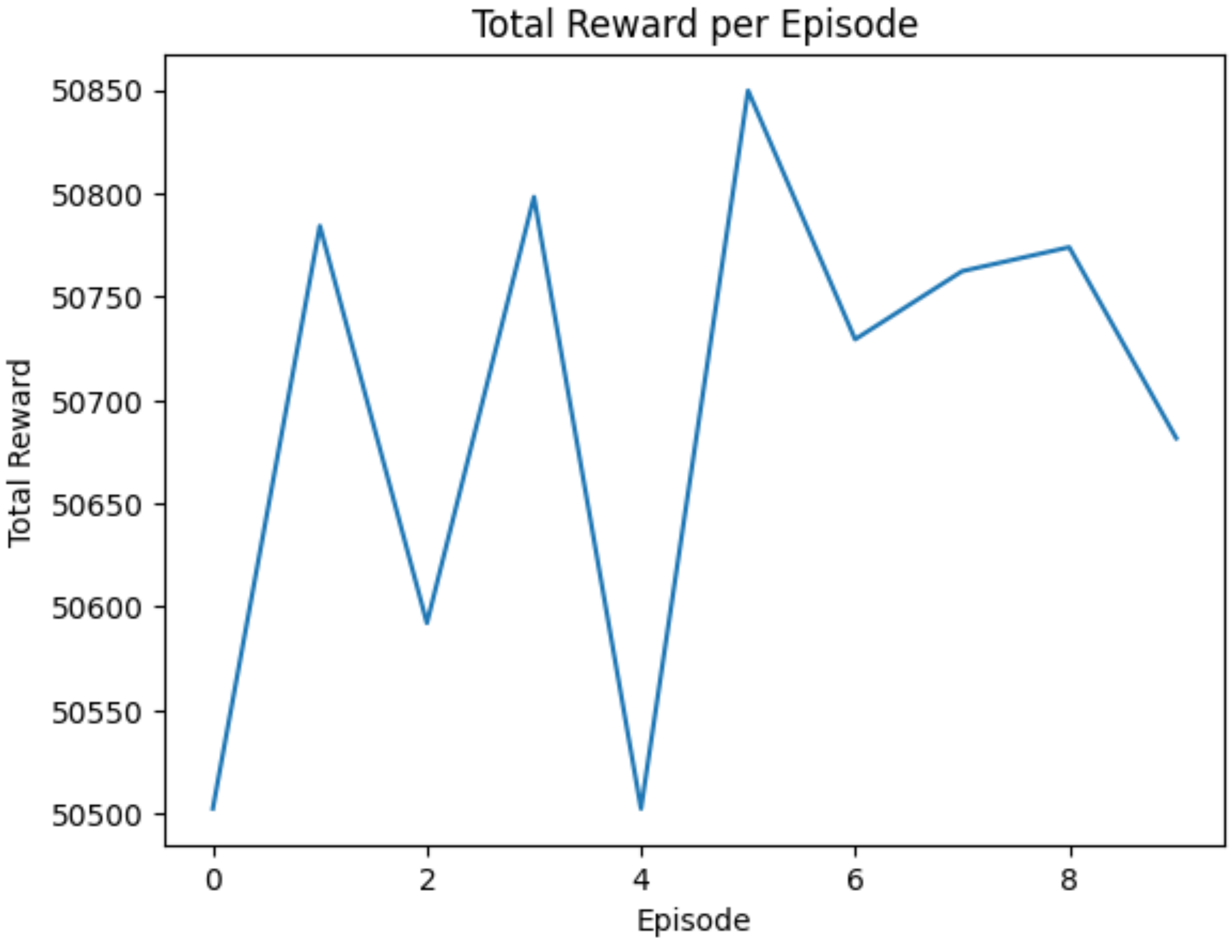
```
|    n_updates            | 10           |
|    policy_gradient_loss | -0.0555      |
|    value_loss           | 3.14         |
-----------------------------------------
-----------------------------------------
| time/                   |              |
|    fps                  | 120          |
|    iterations           | 3            |
|    time_elapsed         | 51           |
|    total_timesteps      | 6144         |
| train/                  |              |
|    approx_kl            | 0.020067055  |
|    clip_fraction        | 0.35         |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.616       |
|    explained_variance   | -0.0751      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.96         |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0443     |
|    value_loss           | 2.48        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 116         |
|    iterations           | 4           |
|    time_elapsed         | 70          |
|    total_timesteps      | 8192        |
| train/                  |             |
|    approx_kl            | 0.023410397 |
|    clip_fraction        | 0.253       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.534      |
|    explained_variance   | -0.116      |
|    learning_rate        | 0.0003      |
|    loss                 | 0.857       |
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0406     |
|    value_loss           | 2.55        |
-----------------------------------------
-----------------------------------------
| time/                   |             |
|    fps                  | 123         |
|    iterations           | 5           |
|    time_elapsed         | 82          |
|    total_timesteps      | 10240       |
| train/                  |             |
|    approx_kl            | 0.037795052 |
|    clip_fraction        | 0.167       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.387      |
|    explained_variance   | -0.0756     |
|    learning_rate        | 0.0003      |
|    loss                 | 0.981       |
|    n_updates            | 40          |
|    policy_gradient_loss | -0.0344     |
|    value_loss           | 3.08        |
-----------------------------------------
<stable_baselines3.ppo.ppo.PPO at 0x798a09d5d960>
```

```python
# Create the environment using the testing data
test_env = HealthcareEnv(test_data)

# Wrap the environment for evaluation
test_env = DummyVecEnv([lambda: HealthcareEnv(test_data)])

# Reset the environment
obs = test_env.reset()

# Evaluate the model for a few episodes
total_rewards = 0
num_episodes = 10

for episode in range(num_episodes):
    obs = test_env.reset()
    episode_reward = 0
    done = False

    while not done:
        action, _states = model.predict(obs, deterministic=True)
        obs, reward, done, info = test_env.step(action)
        episode_reward += reward

    print(f"Episode {episode + 1}: Total Reward = {episode_reward}")
    total_rewards += episode_reward

avg_reward = total_rewards / num_episodes
print(f"\nAverage Reward over {num_episodes} episodes: {avg_reward}")
```

```
Episode 1: Total Reward = [9956.234]
Episode 2: Total Reward = [10027.319]
Episode 3: Total Reward = [10112.456]
Episode 4: Total Reward = [10100.748]
Episode 5: Total Reward = [10098.62]
Episode 6: Total Reward = [10084.622]
Episode 7: Total Reward = [10097.509]
Episode 8: Total Reward = [10072.049]
Episode 9: Total Reward = [10129.446]
Episode 10: Total Reward = [10057.532]

Average Reward over 10 episodes: [10073.653]
```

```
np.random.seed(42)
test_data = pd.DataFrame(np.random.rand(100, 10), columns=[f'feature_{i}' for i

# Use this synthetic data as your test dataset
test_data = (test_data - test_data.min()) / (test_data.max() - test_data.min())
```

```
import matplotlib.pyplot as plt

# Plot rewards over episodes
plt.plot(rewards)
plt.title("Total Reward per Episode")
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.show()
```



```
model = PPO('MlpPolicy', train_env, learning_rate=0.0001, gamma=0.99, n_steps=2
model.learn(total_timesteps=50000)
```

```
|    clip_fraction         | 0.00137      |
|    clip_range           | 0.2           |
|    entropy_loss         | -0.0164       |
|    explained_variance   | -9.06e-06     |
|    learning_rate        | 0.0001        |
|    loss                 | 0.328         |
|    n_updates            | 210           |
|    policy_gradient_loss | -0.000612     |
|    value_loss           | 0.824         |
-------------------------------------------
-------------------------------------------
| time/                   |               |
|    fps                  | 157           |
|    iterations           | 23            |
|    time_elapsed         | 299           |
|    total_timesteps      | 47104         |
| train/                  |               |
|    approx_kl            | 1.6972219e-05 |
|    clip_fraction        | 0.000342      |
|    clip_range           | 0.2           |
|    entropy_loss         | -0.0131       |
|    explained_variance   | 7.63e-06      |
|    learning_rate        | 0.0001        |
|    loss                 | 0.354         |
|    n_updates            | 220           |
|    policy_gradient_loss | -0.000242     |
|    value_loss           | 0.846         |
-------------------------------------------
------------------------------------------
| time/                   |              |
|    fps                  | 157          |
|    iterations           | 24           |
|    time_elapsed         | 311          |
|    total_timesteps      | 49152        |
| train/                  |              |
|    approx_kl            | 8.688058e-05 |
|    clip_fraction        | 0.0021       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.00997     |
|    explained_variance   | 2.92e-06     |
|    learning_rate        | 0.0001       |
|    loss                 | 0.464        |
|    n_updates            | 230          |
|    policy_gradient_loss | -0.000701    |
|    value_loss           | 1.04         |
------------------------------------------
-------------------------------------------
| time/                   |               |
|    fps                  | 156           |
|    iterations           | 25            |
|    time_elapsed         | 327           |
|    total_timesteps      | 51200         |
| train/                  |               |
|    approx_kl            | 1.7708167e-05 |
|    clip_fraction        | 0.000391      |
|    clip_range           | 0.2           |
|    entropy_loss         | -0.0079       |
|    explained_variance   | -1.29e-05     |
```

```
def random_agent(env, num_episodes=10):
    total_rewards = 0
    for episode in range(num_episodes):
        obs = env.reset()
        episode_reward = 0
        done = False
        while not done:
            # Take random actions in batch form for DummyVecEnv
            action = [env.action_space.sample()]  # Wrapping action in a list f
            obs, reward, done, _ = env.step(action)  # Perform action in batch
            episode_reward += reward
        print(f"Episode {episode + 1}: Total Reward = {episode_reward}")
        total_rewards += episode_reward

    avg_reward = total_rewards / num_episodes
    print(f"\nAverage Reward for Random Agent: {avg_reward}")
    return avg_reward

# Evaluate the random agent
random_agent(test_env)
```

```
Episode 1: Total Reward = [-125.85617]
Episode 2: Total Reward = [7.6937885]
Episode 3: Total Reward = [-21.145048]
Episode 4: Total Reward = [169.25829]
Episode 5: Total Reward = [-90.26116]
Episode 6: Total Reward = [-47.93909]
Episode 7: Total Reward = [41.252026]
Episode 8: Total Reward = [-37.584785]
Episode 9: Total Reward = [-4.1457086]
Episode 10: Total Reward = [53.779324]

Average Reward for Random Agent: [-5.4948545]
array([-5.4948545], dtype=float32)
```

```
pip install shap
```

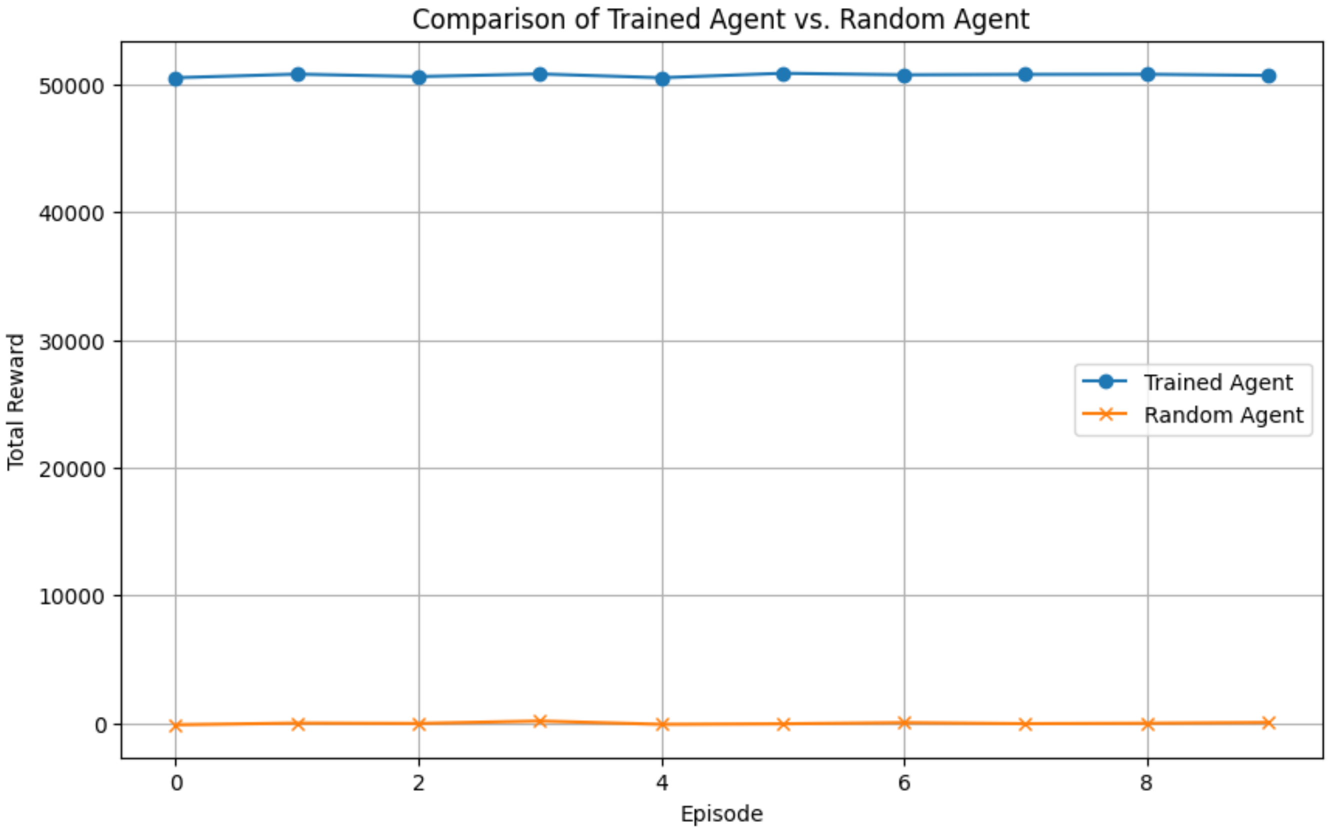
```
Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/di
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.10/d
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dis
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
```

```
# Check the shape of SHAP values
print(f"SHAP values shape: {shap_values.shape}")
```

```
SHAP values shape: (1, 2587)
```

```
import matplotlib.pyplot as plt

# Assuming you have the rewards for the trained agent and the random agent
trained_agent_rewards = [50502.418, 50784.35, 50592.164, 50798.4, 50502.348, 50
random_agent_rewards = [-125.85617, 7.6937885, -21.145048, 169.25829, -90.26116


# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(trained_agent_rewards, label="Trained Agent", marker="o")
plt.plot(random_agent_rewards, label="Random Agent", marker="x")
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.title("Comparison of Trained Agent vs. Random Agent")
plt.legend()
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipykernel.py:283: Deprecat
  and should_run_async(code)
```



Start coding or generate with AI.