Codsoft

ARTIFICIAL INTELLIGENCE PROJECT

Name- Sudipta Samanta

TASK 2: TIC-TAC-TOE AI

Implement an AI agent that plays the classic game of Tic-Tac-Toe against a human player. You can use algorithms like Minimax with or without Alpha-Beta Pruning to make the AI player unbeatable. This project will help you understand game theory and basic search algorithms

Source Code:

```
BOARD_EMPTY = 0
PLAYER_X = 1
PLAYER_O = -1

def print_board(s):
  def convert(num):
    if num == PLAYER_X:
      return 'X'
    if num == PLAYER_O:
      return 'O'
    return '_'

  i = 0
  for _ in range(3):
    for _ in range(3):
      print(convert(s[i]), end=' ')
      i += 1
    print()

from collections import Counter

def player(s):
```

```
  counter = Counter(s)
  x_places = counter[1]
  o_places = counter[-1]

  if x_places + o_places == 9:
    return None
  elif x_places > o_places:
    return PLAYER_O
  else:
    return PLAYER_X

def actions(s):
  play = player(s)
  actions_list = [(play, i) for i in range(len(s)) if s[i] == BOARD_EMPTY]
  return actions_list

def result(s, a):
  (play, index) = a
  s_copy = s.copy()
  s_copy[index] = play
  return s_copy

def terminal(s):
  for i in range(3):
    # Checking if a row is filled and equal.
    if s[3 * i] == s[3 * i + 1] == s[3 * i + 2] != BOARD_EMPTY:
      return s[3 * i]
    # Checking if a column is filled and equal.
    if s[i] == s[i + 3] == s[i + 6] != BOARD_EMPTY:
      return s[i]

  # Checking if a diagonal is filled and equal.
  if s[0] == s[4] == s[8] != BOARD_EMPTY:
    return s[0]
  if s[2] == s[4] == s[6] != BOARD_EMPTY:
    return s[2]

  # Checking if the game has no more moves available
  if player(s) is None:
    return 0
```

```
  # Return None if none of the previous conditions satisfy.
  return None

def utility(s):
  term = terminal(s)
  # Return who wins the game if the game has terminated
  if term is not None:
    return term

  # Get the list of actions available
  action_list = actions(s)
  utils = []
  for action in action_list:
    # Create a new state applying the action to current state
    new_s = result(s, action)
    # Add the score of the new state to a list
    utils.append(utility(new_s))

  score = utils[0]
  play = player(s)
  # Calculate the max score if X is playing
  if play == PLAYER_X:
    for i in range(len(utils)):
      if utils[i] > score:
        score = utils[i]
  # Calculate the min score if O is playing
  else:
    for i in range(len(utils)):
      if utils[i] < score:
        score = utils[i]
  return score

def utility(s, cost):
  term = terminal(s)
  if term is not None:
    # Return the cost of reaching the terminal state
    return (term, cost)

  action_list = actions(s)
```

```
  utils = []
  for action in action_list:
    new_s = result(s, action)
    # Every recursion will be an increment in cost
    utils.append(utility(new_s, cost + 1))

  # Remember the associated cost with the score of the state.
  score = utils[0][0]
  idx_cost = utils[0][1]
  play = player(s)
  if play == PLAYER_X:
    for i in range(len(utils)):
     if utils[i][0] > score:
       score = utils[i][0]
       idx_cost = utils[i][1]
  else:
    for i in range(len(utils)):
      if utils[i][0] < score:
        score = utils[i][0]
        idx_cost = utils[i][1]

  # Return the score with the associated cost.
  return (score, idx_cost)

def minimax(s):
  action_list = actions(s)
  utils = []
  for action in action_list:
    new_s = result(s, action)
    utils.append((action, utility(new_s, 1)))
  # the score and "cost" of that action.

  if len(utils) == 0:
    return ((0, 0), (0, 0))

  # Sort the list in ascending order of cost.
  sorted_list = sorted(utils, key=lambda l : l[0][1])
  # Since the computer shall be Player O,
  # It is safe to return the object with minimum score.
  action = min(sorted_list, key = lambda l : l[1])
```

```
  return action

if __name__ == '__main__':
  # Initializing the state
  s = [BOARD_EMPTY for _ in range(9)]
  print('|------- Welcome to Tic Tac Toe! -----------|')
  print('You are X while the Computer is O. Lets play!\n')

  # Run the program while the game is not terminated
  while terminal(s) is None:
    play = player(s)
    if play == PLAYER_X:
      # Take input from user
      print('\n\nIt is your turn', end='\n\n')
      x = int(input('Enter the x-coordinate [0-2]: '))
      y = int(input('Enter the y-coordinate [0-2]: '))
      index = 3 * x + y

      if not s[index] == BOARD_EMPTY:
        print('Oops! That coordinate is already taken. Try again.\n')
        continue

      # Apply the action and print the board
      s = result(s, (PLAYER_X, index))
      print_board(s)
    else:
      print('\n\nThe is computer is playing its turn')
      # Get the action by running the minimax algorithm
      action = minimax(s)
      # Apply the returned action to the state and print the board
      s = result(s, action[0])
      print_board(s)

  # determine the winner
  winner = terminal(s)
  if winner == PLAYER_X:
    print("You have won!")
  elif winner == PLAYER_O:
    print("You have lost!")
  else:
```

```
print("It's a tie.")
```

Output:

```
|------- Welcome to Tic Tac Toe! -----------|
You are X while the Computer is O. Lets play!



It is your turn

Enter the x-coordinate [0-2]: 1
Enter the y-coordinate [0-2]: 1
_ _ _
_ X _
_ _ _


The is computer is playing its turn
O _ _
_ X _
_ _ _


It is your turn

Enter the x-coordinate [0-2]: 2
Enter the y-coordinate [0-2]: 2
O _ _
_ X _
_ _ X
```

```
The is computer is playing its turn
O _ O
_ X _
_ _ X


It is your turn

Enter the x-coordinate [0-2]: 0
Enter the y-coordinate [0-2]: 1
O X O
_ X _
_ _ X


The is computer is playing its turn
O X O
_ X _
_ O X


It is your turn

Enter the x-coordinate [0-2]: 0
Enter the y-coordinate [0-2]: 1
Oops! That coordinate is already taken. Try again.



It is your turn
```

```
Enter the x-coordinate [0-2]: 1
Enter the y-coordinate [0-2]: 0
O X O
X X _
_ O X


The is computer is playing its turn
O X O
X X O
_ O X


It is your turn

Enter the x-coordinate [0-2]: 2
Enter the y-coordinate [0-2]: 0
O X O
X X O
X O X
It's a tie.
```

TASK 1: CHATBOT WITH RULE-BASED RESPONSES

Build a simple chatbot that responds to user inputs based on predefined rules. Use if-else statements or pattern matching techniques to identify user queries and provide appropriate responses. This will give you a basic understanding of natural language processing and conversation flow

Source Code:

```
# Install necessary libraries
!pip install nltk

# Import necessary modules
import nltk
import re
from nltk.chat.util import Chat, reflections
```

```
# Download NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# Define patterns and responses
pairs = [
    [r"My name is (.*)", ["Hello %1, nice to meet you, how can I assist you
today?",]],
    [r"Hi|Hey|Hello", ["Hello, how can I help you?", "Hi! How can I assist you
today?",]],
    [r"What is your name?", ["My name is chatbot created to assist you. what's
on your mind today?",]],
    [r"How are you?", ["I'm good! what about you? ",]],
    [r"Can you help me with (.*)", ["Sure, I can help you with %1. Please provide
more details.",]],
    [r"Thankyou|Thanks", ["You're welcome!", "Happy to help you! Let me know
if you need anything else.",]],
    [r"Bye", ["Bye! Have a nice day!", "Goodbye!",]],
    [r"What is your purpose?", ["I'am here to assist with your queries, provide
information according to the pattern I have been created with!",]],
    [r"(.*)", ["I'm sorry, I don't understand that. Could you clarify?",]],
]

# Define the chatbot class
class RuleBasedChatbot:
    def __init__(self, pairs):
        self.chat = Chat(pairs, reflections)

    def respond(self, user_input):
        return self.chat.respond(user_input)

# Initialize the chatbot
chatbot = RuleBasedChatbot(pairs)

# Function to chat with the bot
def chat_with_bot():
    print("Hi, I'm your chatbot. Type 'exit' to exit chat with chatbot.")
    while True:
        user_input = input("You: ")
```

```
    if user_input.lower() == 'exit':
        print("Chatbot: Bye! Have a great day!")
        break
    response = chatbot.respond(user_input)
    print(f"Chatbot: {response}")

# Start chatting with the bot
chat_with_bot()
```

Output:

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
Hi, I'm your chatbot. Type 'exit' to exit chat with chatbot.
You: Hello
Chatbot: Hi! How can I assist you today?
You: What is your purpose?
Chatbot: I'am here to assist with your queries, provide information according to the pattern I have been created with!
You: Thanks
Chatbot: Happy to help you! Let me know if you need anything else.
You: Bye
Chatbot: Goodbye!
You: exit
Chatbot: Bye! Have a great day!
```

✓ 2m 12s completed at 22:05

TASK 4: RECOMMENDATION SYSTEM

Create a simple recommendation system that suggests items to users based on their preferences. You can use techniques like collaborative filtering or content-based filtering to recommend movies, books, or products to users.

Source Code:

Cell1

```
#Import all necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.html.widgets import *
sns.set_style('white')
%matplotlib inline
```

Cell2

```
#Get the data into Pandas Dataframe object
import pandas as pd
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('dataset.csv', sep = '\t', names =
column_names)
df.head()
```

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

Cell3

```
#Get the Movie Titles
import pandas as pd
movie_titles = pd.read_csv('movieIdTitles.csv')
movie_titles.head()
```

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

Cell4

```
#Merge the dataset with movie titles
df = pd.merge(df, movie_titles, on = 'item_id')
df.head()
```

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	0	172	5	881250949	Empire Strikes Back, The (1980)
2	0	133	1	881250949	Gone with the Wind (1939)
3	196	242	3	881250949	Kolya (1996)
4	186	302	3	891717742	L.A. Confidential (1997)

Cell5

```
df.groupby('title')['rating'].mean().sort_values(ascending =
False).head()
```

title	rating
They Made Me a Criminal (1939)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Saint of Fort Washington, The (1993)	5.0
Someone Else's America (1995)	5.0
Star Kid (1997)	5.0

dtype: float64

Cell6

```
df.groupby('title')['rating'].count().sort_values(ascending = False).head()
```

title	rating
Star Wars (1977)	584
Contact (1997)	509
Fargo (1996)	508
Return of the Jedi (1983)	507
Liar Liar (1997)	485

dtype: int64

Cell7

```
ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()
```

title	rating
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

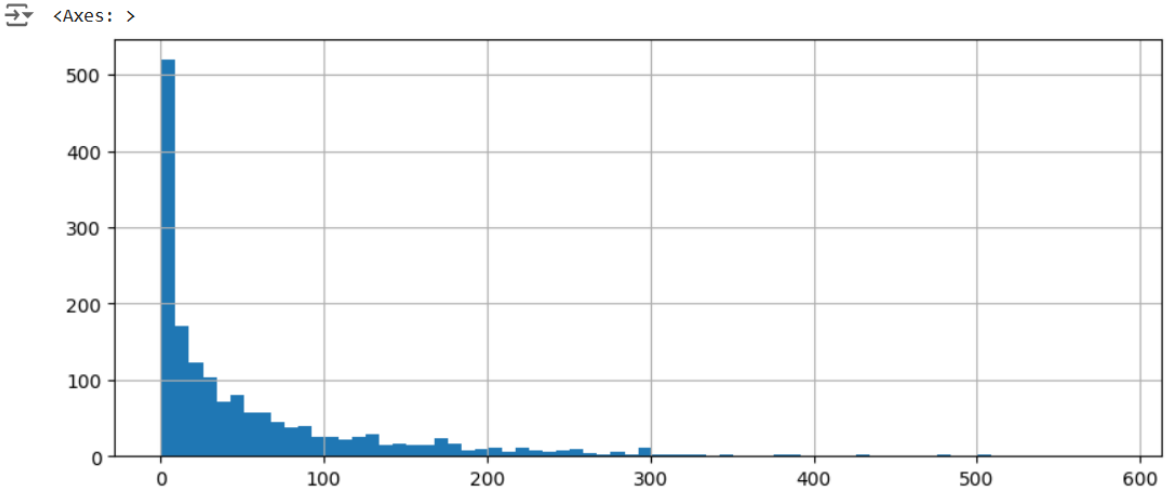
Cell8

```
ratings['numOfRatings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

title	rating	numOfRatings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

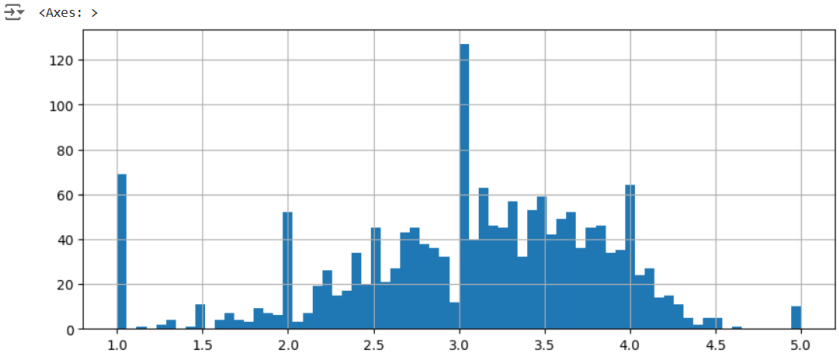
Cell9

```
import matplotlib.pyplot as plt
plt.figure(figsize = (10,4))
ratings['numOfRatings'].hist(bins = 70)
```



Cell10

```
plt.figure(figsize = (10,4))
ratings['rating'].hist(bins = 70)
```



Cell11

```
moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Young Poisoner's Handbook, The (1995)	Zeus and Roxanne (1997)	unknown
user_id																				
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	NaN	NaN	4.0
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 1664 columns

Cell12

```
#Most Rated Movies with their Average Ratings
ratings.sort_values('numOfRatings', ascending = False).head(10)
```

title	rating	numOfRatings
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

Cell13

```
for i in ratings.index:
    movieUserRatings = moviemat[i]
```

```
    similarToThatMovie = moviemat.corrwith(movieUserRatings)
    corr_toMovie = pd.DataFrame(similarToThatMovie, columns = ['Correlation'])
    corr_toMovie.dropna(inplace = True)
    corr_toMovie = corr_toMovie.join(ratings['numOfRatings'])
    result = corr_toMovie[corr_toMovie['numOfRatings'] > 100].sort_values('Correlation', ascending = False).head()
    if result['numOfRatings'].count() >= 5:
        print(i)
        ratings.loc[i, 'FirstMovieRecommendation'] = result.iloc[1:2].index.values[0]
        ratings.loc[i, 'SecondMovieRecommendation'] = result.iloc[2:3].index.values[0]
        ratings.loc[i, 'ThirdMovieRecommendation'] = result.iloc[3:4].index.values[0]
        ratings.loc[i, 'FourthMovieRecommendation'] = result.iloc[4:5].index.values[0]
```

```
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2897: RuntimeWarning: invalid value encountered in divide
  c /= stddev[:, None]
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2898: RuntimeWarning: invalid value encountered in divide
  c /= stddev[None, :]
'Til There Was You (1997)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2897: RuntimeWarning: invalid value encountered in divide
  c /= stddev[:, None]
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2898: RuntimeWarning: invalid value encountered in divide
  c /= stddev[None, :]
1-900 (1994)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2897: RuntimeWarning: invalid value encountered in divide
```

Cell14

```
#Check the result
ratings.head()
```

Cell15

```
ratings = ratings.fillna('-')
```

Cell16

```
#Save the ratings data for later use
ratings.to_csv('MovieRecommendations.csv', encoding='utf-
8')
```

Cell17

```
#Load the dataset saved for reusability from this code block
onwards
df_result = pd.read_csv('MovieRecommendations.csv')
df_result.head()
```

Cell18

```
import ipywidgets as widgets
from IPython.display import display
inputMovieName = widgets.Text()

def getRecommendations(sender):
    searchMovie = inputMovieName.value
    list_result = df_result[df_result['title'] == searchMovie]
    fm = list_result['FirstMovieRecommendation'].values[0]
    sm = list_result['SecondMovieRecommendation'].values[0]
    tm = list_result['ThirdMovieRecommendation'].values[0]
    fourthm =
list_result['FourthMovieRecommendation'].values[0]
    finalRecommendationText = '1:' + fm + ' \n2:' + sm + ' \n3:'
+ tm + ' \n4:' + fourthm
    print('Your Recommendations for the Movie ' +
searchMovie + ' are:\n')
    print(finalRecommendationText)
```