EXPERIMENT NO. 11

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-V – AIML-V-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL347-AAIES-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 11.11.2023
Faculty Signature:
Grade:

Objective(s):

- Study Expert System.
- Understand and implement simple backwards chaining expert system.

Outcome:

Students will be familiarized with Expert System.

Problem Statement:

Develop a Restaurant Decision Assistant using simple backward chaining in Python. The assistant will help users decide on a suitable restaurant based on their preferences and dietary restrictions. The system will use backward chaining to infer the user's desired type of cuisine and dietary requirements by asking a series of questions.

Rules are:

1. If the user prefers spicy food, the system will suggest restaurants that offer cuisines like Indian, Thai, or Mexican, known for their spicy dishes.
2. If the user does not prefer spicy food, the system will suggest restaurants that serve milder cuisines such as Italian or American.
3. If the user follows a vegetarian diet, the system will recommend restaurants with a variety of vegetarian options, ensuring a satisfying dining experience.
4. If the user follows a vegan diet, the system will prioritize restaurants that offer vegan-friendly menus to cater to their dietary preferences.
5. If the user has specific dietary restrictions or allergies, the system will consider those limitations and suggest restaurants with suitable menu items, such as gluten-free or lactose-free options.
6. Based on the user's preference for a casual or fine-dining experience, the system will recommend restaurants that match the desired ambiance and style of dining.
7. The system may also consider the user's location to suggest nearby restaurants, making dining choices more convenient and accessible.
8. After considering all the user's responses, the system will provide a final restaurant recommendation that aligns with their inferred preferences and dietary needs.

Background Study:

Expert System is an interactive and reliable computer-based decision-making system which uses both facts and heuristics to solve complex decision-making problems. It is considered at the highest level of human intelligence and expertise. The purpose of an expert system is to solve the most complex issues in a specific domain.

Simple backward chaining is a basic inference technique used in expert systems to reach conclusions based on a set of predefined rules. It starts with the user's goal or query and works

backward through the rules to find supporting facts until the system reaches a known fact or an initial assumption. By examining the facts in reverse, the system deduces the necessary conditions to satisfy the user's goal.

Question Bank:

1. What is Expert System? State some examples.
   An expert system is a computer-based AI system that emulates the decision-making abilities of a human expert in a specific domain. It uses knowledge, rules, and inference mechanisms to provide solutions, recommendations, or explanations in a particular field. Examples include medical diagnosis systems like MYCIN, customer support chatbots, and financial advisory systems.

2. What are characteristics of Expert System?
   - Knowledge Base: Contains domain-specific information, rules, and facts.
   - Inference Engine: Processes the knowledge and applies rules to draw conclusions or make recommendations.
   - User Interface: Allows interaction between users and the system, receiving queries and presenting results.
   - Explanation Facility: Explains the reasoning behind the system's recommendations.
   - Knowledge Acquisition: Methods for extracting and inputting knowledge into the system.
   - Domain Expertise: Focuses on a specific domain, often limited to narrow expertise.

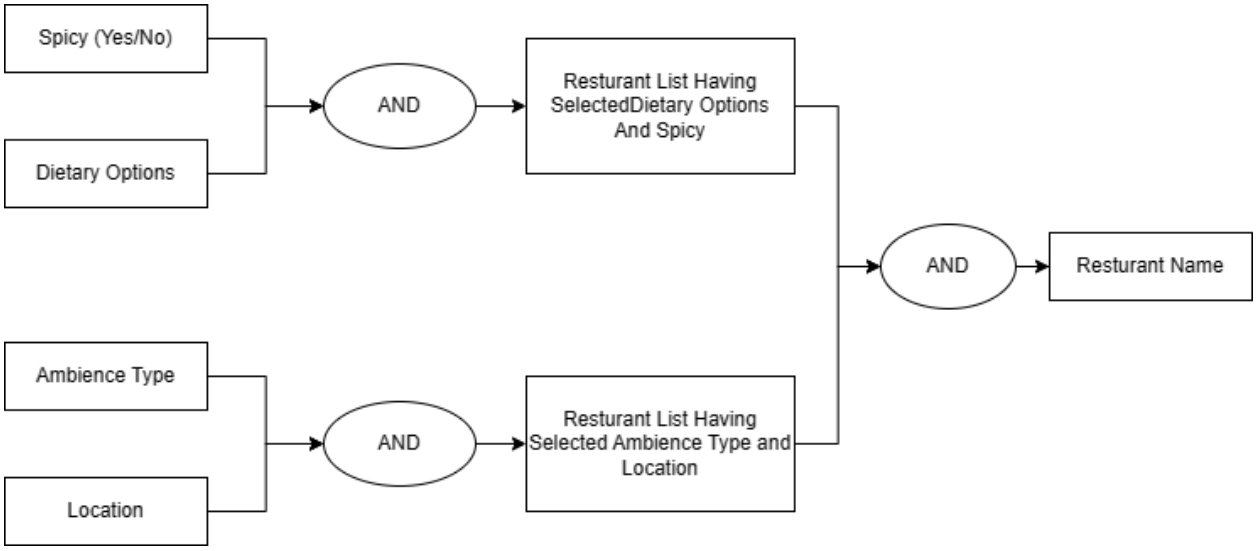
3. State limitations and advantages of Expert system?
   Advantages:
   - **Consistency**: Provides consistent and uniform decisions.
   - **Availability**: Offers 24/7 availability for consultations and assistance.
   - **Scalability**: Can store vast amounts of knowledge and serve many users.
   - **Capture Expertise**: Preserves and shares expert knowledge.
   - **Decision Transparency**: Provides explanations for its decisions, aiding user understanding.

   Limitations:
   - **Narrow Focus**: Limited to the domain they are designed for.
   - **Knowledge Acquisition**: Gathering accurate and relevant knowledge can be challenging.
   - **Maintenance**: Regular updates are needed to keep the system's knowledge current.
   - **Lack of Common Sense**: May lack human-like understanding and common-sense reasoning.
   - **Complexity**: Developing advanced systems can be time-consuming and expensive.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Flowchart



Code:

Experiment 11

Problem Link:

Develop a Restaurant Decision Assistant using simple backward chaining in Python. The assistant will help users decide on a suitable restaurant based on their preferences and dietary restrictions. The system will use backward chaining to infer the user's desired type of cuisine and dietary requirements by asking a series of questions.

Name	Cuisine	Dietary Options	Ambiance	Location
Spice Delight	Indian, Thai, Mexican	Vegetarian, Vegan	Casual	Downtown
Mild Flavors	Italian, American	Vegetarian	Casual	Midtown
Veggie Haven	Vegetarian, Vegan	Gluten-free, Lactose-free	Casual	Suburb
Fine Dining Delights	Italian, French, Japanese	Vegetarian	Fine Dining	Uptown
Healthy Bites	Mediterranean, Salads	Vegetarian, Vegan, Gluten-free	Casual	Downtown
Tasty Treats	American, Mexican	Vegetarian	Casual	Midtown
Vegan Delights	Vegan	Gluten-free, Lactose-free	Casual	Suburb

Rules are:

1. If the user prefers spicy food, the system will suggest restaurants that offer cuisines like Indian, Thai, or Mexican, known for their spicy dishes.
2. If the user does not prefer spicy food, the system will suggest restaurants that serve milder cuisines such as Italian or American.
3. If the user follows a vegetarian diet, the system will recommend restaurants with a variety of vegetarian options, ensuring a satisfying dining experience.
4. If the user follows a vegan diet, the system will prioritize restaurants that offer vegan-friendly menus to cater to their dietary preferences.
5. If the user has specific dietary restrictions or allergies, the system will consider those limitations and suggest restaurants with suitable menu items, such as gluten-free or lactose-free options.
6. Based on the user's preference for a casual or fine-dining experience, the system will recommend restaurants that match the desired ambiance and style of dining.
7. The system may also consider the user's location to suggest nearby restaurants, making dining choices more convenient and accessible.
8. After considering all the user's responses, the system will provide a final restaurant recommendation that aligns with their inferred preferences and dietary needs.

Code:

[21]
```python
# importing required libraries
import numpy as np
import pandas as pd
```
Python

```python
# function to create a dataframe of the given data
def create_dataframe():
    data = {'Name': ['Spice Delight', 'Mild Flavors', 'Veggie Haven', 'Fine Dining Delights', 'Healthy Bites', 'Tasty Treats', 'Vegan Delights'],
            'Cuisine': ['Indian, Thai, Mexican', 'Italian, American', 'Vegetarian, Vegan', 'Italian, French, Japanese', 'Mediterranean, Salads', 'American,
            'Dietary Options': ['Vegetarian, Vegan', 'Vegetarian', 'Gluten-free, Lactose-free', 'Vegetarian', 'Vegetarian, Vegan, Gluten-free', 'Vegetarian'
            'Ambiance': ['Casual', 'Casual', 'Casual', 'Fine Dining', 'Casual', 'Casual', 'Casual'],
            'Location': ['Downtown', 'Midtown', 'Suburb', 'Uptown', 'Downtown', 'Midtown', 'Suburb']}
    df = pd.DataFrame(data)
    return df
```

[22] Python

```python
# function to get user preferences for cuisine and dietary requirements.
def get_user_preferences():
    # dictionary to hold user preferences
    preferences = {}

    # Rule 1: Dietary preferences
    print("Do you follow a specific diet?")
    print("1. Vegetarian")
    print("2. Vegan")
    print("3. Gluten-free")
    print("4. Lactose-free")
    print("5. None")
    dietary = int(input("Enter your preference (number): "))
    preferences['dietary'] = dietary

    # Rule 2: Ambiance preference
    print("What kind of ambiance do you prefer?")
    print("1. Casual")
    print("2. Fine Dining")
    ambiance = int(input("Enter your preference (number): "))
    preferences['ambiance'] = ambiance

    # Rule 3: Location preference
    print("In which location do you prefer to dine?")
    print("1. Downtown")
    print("2. Midtown")
    print("3. Suburb")
    print("4. No preference")
    location = int(input("Enter your preference (number): "))
    preferences['location'] = location

    # Rule 4: Cuisine preference
    print("Do you prefer spicy food?")
    print("1. Yes")
    print("2. No")
    cuisine = int(input("Enter your preference (number): "))
    preferences['cuisine'] = cuisine

    return preferences
```


[23] Python

```python
# function to recommend a restaurant based on the dataset and user preferences.
def restaurant_recommendation(df, preferences):
    # filter based on dietary preferences
    if preferences['dietary'] == 1:  # Vegetarian
        df = df[df['Dietary Options'].str.contains('Vegetarian')]
    elif preferences['dietary'] == 2:  # Vegan
        df = df[df['Dietary Options'].str.contains('Vegan')]
    elif preferences['dietary'] == 3:  # Gluten-free
        df = df[df['Dietary Options'].str.contains('Gluten-free')]
    elif preferences['dietary'] == 4:  # Lactose-free
        df = df[df['Dietary Options'].str.contains('Lactose-free')]

    # filter based on ambiance preferences
    if preferences['ambiance'] == 1:
        df = df[df['Ambiance'] == 'Casual']
    elif preferences['ambiance'] == 2:
        df = df[df['Ambiance'] == 'Fine Dining']

    # filter based on location preferences
    if preferences['location'] == 1:
        df = df[df['Location'] == 'Downtown']
    elif preferences['location'] == 2:
        df = df[df['Location'] == 'Midtown']
    elif preferences['location'] == 3:
        df = df[df['Location'] == 'Suburb']

    # filter based on cuisine preferences
    if preferences['cuisine'] == 1:
        df = df[df['Cuisine'].str.contains('Indian|Thai|Mexican')]
    elif preferences['cuisine'] == 2:
        df = df[~df['Cuisine'].str.contains('Indian|Thai|Mexican')]

    # if there are no matching restaurants, return None
    if df.empty:
        return None

    # return the first recommendation
    return df.iloc[0]['Name']
```

[24] Python

```python
df = create_dataframe()
simulated_preferences = get_user_preferences()
recommendation = restaurant_recommendation(df, simulated_preferences)

# Display the recommendation
if recommendation:
    print("Recommended restaurant based on simulated preferences:", recommendation)
else:
    print("Sorry, no matching restaurants found based on simulated preferences.")
```

[25] Python

```
Do you follow a specific diet?
1. Vegetarian
2. Vegan
3. Gluten-free
4. Lactose-free
5. None
Enter your preference (number): 3
What kind of ambiance do you prefer?
1. Casual
2. Fine Dining
Enter your preference (number): 1
In which location do you prefer to dine?
1. Downtown
2. Midtown
3. Suburb
4. No preference
Enter your preference (number): 2
Do you prefer spicy food?
1. Yes
2. No
Enter your preference (number): 2
Sorry, no matching restaurants found based on simulated preferences.
```