Applied AI and Expert Systems

Lab Manual

Department of Computer Science and Engineering
The NorthCap University, Gurugram

Applied AI and Expert Systems

CSL 347

Er. Shivam Shivam

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2023-2024

Published by:

School of Engineering and Technology

Department of Computer Science & Engineering

The NorthCap University Gurugram

• Laboratory Manual is for Internal Circulation only

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

PREFACE

Applied AI and Expert Systems Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of "teach yourself pattern" and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The pre-requisite is having a basic working knowledge of Python. The laboratory exercises will include familiarization with Rule-based systems, search algorithms like Breadth First Search, Depth First Search, Heuristic and optimization based algorithms, graph visualizations and knowledge graph representations, expert systems, machine learning, deep learning algorithms and generative AI based large language models. Students would learn the algorithms pertaining to these and implement the same using a high-level language, i.e. Python. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the The NorthCap University, Gurugram. Teacher's copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE, IT, ECE and BSc branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

Authors
The NorthCap University
Gurugram, India

THE NORTHCAP UNIVERSITY

CONTENTS

1. INTRODUCTION

That 'learning is a continuous process' cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus, practical makes an integral part of a learning process.

OBJECTIVE:
The purpose of conducting experiments can be stated as follows:

- To familiarize the students with the basic concepts of with search algorithms like Breadth First Search, Depth First Search, Hill climbing etc., Game Playing Algorithms, semantic networks, fuzzy sets and expert systems.
- The lab sessions will be based on exploring the concepts discussed in class.
- Learning and understanding Search Algorithms.
- Learning and understanding Heuristic and optimization-based Algorithms.
- Learning and understanding Expert Systems.
- Learning and understanding Machine learning and Deep Learning models.
- Learning and understanding Generative AI with large language models.
- Hands on experience.

2. LAB REQUIREMENTS

S.No.	Requirements	Details
1	**Software Requirements**	Python 3.
2	**Operating System**	Windows(64-bit), Linux
3	**Hardware Requirements**	8 GB RAM (Recommended) 2.60 GHz (Recommended)
4	**Required Bandwidth**	NA

3. GENERAL INSTRUCTIONS

3.1 General discipline in the lab

- Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
- Students will not be allowed to enter late in the lab.
- Students will not leave the class till the period is over.
- Students should come prepared for their experiment.
- Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
- Students must get the connection of the hardware setup verified before switching on the power supply.
- Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
- Violating the above code of conduct may attract disciplinary action.
- Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

3.2 Attendance

- Attendance in the lab class is compulsory.
- Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
- On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

3.3 Preparation and Performance

- Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with

information about self-study reference is provided on LMS.

- Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
- Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
- Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
- Refer **Annexure 1** for Lab Report Format

4. LIST OF EXPERIMENTS

Sr. No.	Title of the Experiment	Software used	Unit Covered	CO Covered
1.	Develop a rule-based pet care assistant in Python with the purpose of providing users with helpful reminders and suggestions for pet care.	Python (Jupyter)	1	CO1
2.	Implement Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms in Python to analyze a simple social network.	Python (Jupyter)	2	CO2
3.	Write a basic state space search program in Python to solve the classic "Missionaries and Cannibals" puzzle.	Python (Jupyter)	2	CO2
4.	Python program for budget allocation of a company using Greedy Algorithm.	Python (Jupyter)	2	CO2
5.	Write a Python program to implement Travelling Salesman problem using Branch and Bound.	Python (Jupyter)	2	CO2
6.	Form triples from a given paragraph. and create knowledge graph using python.	Python (Jupyter)	3	CO3
7.	Using Bayesian network, implement a Python program to calculate probability in disease diagnosis.	Python (Jupyter)	3	CO3
8.	Python program for graph-based visualization and logical reasoning to identify most efficient delivery routes.	Python (Jupyter)	3	CO3
9.	Python code for an expert system to identify potential quality problems.	Python (Jupyter)	4	CO4
10.	Python program to build an Expert based system (similar to MYCIN and DART) for Plant Identification.	Python (Jupyter)	4	CO4
11.	Develop a Restaurant Decision Assistant using simple backward chaining in Python	Python (Jupyter)	4	CO4
12.	Python code to clean and visualize a dataset containing information about a Student's Exam Details.	Python (Jupyter)	5	CO5
13.	Python program to implement Credit Card Fraud detection using Support Vector Machine classification.	Python (Jupyter)	5	CO5

14.	Python program to gender recognition from Facial Images using Convolutional Neural Network.	Python (Jupyter)	5	CO5
15.	To implement simple PDF Document search using Open Source Generative AI model.	Python (Jupyter)	5	CO5

5. LIST OF FLIP EXPERIMENTS

5.1 Project – Machine Learning and Neural Network based models to solve real world problems.
5.2 Project- development of AI based expert system in healthcare sector.
5.3 Competition on Kaggle

6. LIST OF PROJECTS

1. Game of Chess

Chess is a popular game, and in order to improve enjoyment of it, implement a good artificial intelligence system that can compete with humans and make chess a difficult task. Artificial intelligence has changed how top-level chess games are played. The majority of Grandmasters and Super Grandmasters use these latest Artificial Intelligence chess engines to evaluate their own and their opponents' games.

2. Development of AI based expert System for the part- and process specific marking of materials: Due to regulations and industry-specific standards, parts of machines, for example in the health care sector, have to be labelled. However, only few engineering experts possess the knowledge in marking technology within manufacturing companies. Hence, a hybrid expert system which reproduces given problem-solving abilities from experts with regard to their technological and process knowledge during the marketing of material.

3. Chatbots

One of the best AI-based projects is to create a chatbot. You should start by creating a basic chatbot for customer service. You can take inspiration from the chatbots present on various

websites. Once you've created a simple chatbot, you can improve it and create a more detailed version of the same. You should begin by developing a basic customer service chatbot and improve it further.

7. RUBRICS

Marks Distribution	
Continuous Evaluation (50 Marks)	**End semester Exam (20 Marks)**
Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded based on students performance in lab and viva out of total 20.	The project shall be evaluated and at the end of the semester viva will be conducted related to the projects as well as concepts learned in labs.
Following is the breakup of 10 marks for each **4 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment. **3 Marks:** For report writing **3 Marks:** For the 15 minutes quiz to be conducted in every lab.	

Annexure 1

Applied AI and Expert Systems

(CSL 347)

Lab Practical Report

Faculty Name: Ankita Banerjee

Student Name: Piyush Gambhir

Roll No.: 21CSU349

Semester: V

Group: AIML-B (AL-3)

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2023-24

INDEX

S.No	Experiment	Page No.	Date of Experiment	Date of Submission	Marks	CO Covered	Sign

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 1

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: 05.08.2023
Faculty Signature:
Grade:

Objective(s):

- Understand what a rule-based system is.
- Study about how rule-based systems work.
- Design a rule-based system for solving a real-world problem.

Outcome:

Students would be able to design Rule based systems for solving problems.

Problem Statement:

Develop a rule-based pet care assistant in Python with the purpose of providing users with helpful reminders and suggestions for pet care.
The program will offer personalized care tips based on the type of pet, including dogs, cats, and birds, using predefined rules.

Background Study: Rule-based decision-making is a fundamental concept in computer science and artificial intelligence. It involves defining a set of rules or conditions to guide the decision-making process. In the context of the pet care assistant, these rules will be used to determine the type of pet and provide relevant care tips based on the user's selection.

Question Bank:

1. What are some challenges that you faced when creating this rule-based system?
   - Knowledge Acquisition: Gathering accurate and comprehensive rules from experts can be time-consuming and challenging.
   - Rule Complexity: Dealing with a large number of rules and managing their interactions can be complex.
   - Maintenance: Regular updates and adjustments to rules are necessary as knowledge evolves.

- Knowledge Elicitation: Expressing expert knowledge in a machine-readable format can be prone to misinterpretation.
- Handling Exceptions: Dealing with exceptions and handling edge cases within rules can be difficult.

2. What could be some advantages and disadvantages of such Rule based systems?

Advantages:

- **Transparency**: The reasoning behind decisions is explicit and interpretable.
- **Structured Knowledge**: Rules organize domain knowledge in a structured manner.
- **Control**: Experts can directly define and modify rules, providing control over decision-making.
- **Scalability**: Rule-based systems can handle a wide range of cases by adding or modifying rules.

Disadvantages:

- **Limited Context**: May struggle with nuanced decision-making that requires broader context or common sense.
- **Brittleness**: Highly dependent on accurate rules and can fail when rules conflict or are incomplete.
- **Maintenance Overhead**: Regular updates and adjustments to rules can be resource-intensive.
- **Lack of Learning**: Rule-based systems often lack the ability to adapt and learn from new data.

3. How can rule-based systems be made more efficient?

- Rule Pruning: Eliminate redundant or less impactful rules to streamline decision-making.
- Rule Ordering: Arrange rules strategically to prioritize more important conditions.
- Caching and Memoization: Store intermediate results to avoid redundant calculations.
- Fuzzy Logic: Incorporate fuzzy logic to handle imprecise or uncertain information.
- Inference Optimization: Use efficient algorithms for rule matching and inference, like Rete algorithm.
- Machine Learning Integration: Combine rule-based systems with machine learning to learn patterns from data.
- Hybrid Approaches: Integrate rule-based systems with probabilistic or neural network models for enhanced accuracy and robustness.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Approach

1. **get_pet_choice()**: This function displays pet options to the user (Dog, Cat, Parrot) and prompts the user to select a pet by entering a corresponding number. It ensures the user's input is valid and then returns the choice.

2. **get_pet_age()**: This function prompts the user to enter their pet's age in years. It ensures the age is a valid non-negative number and then returns it.

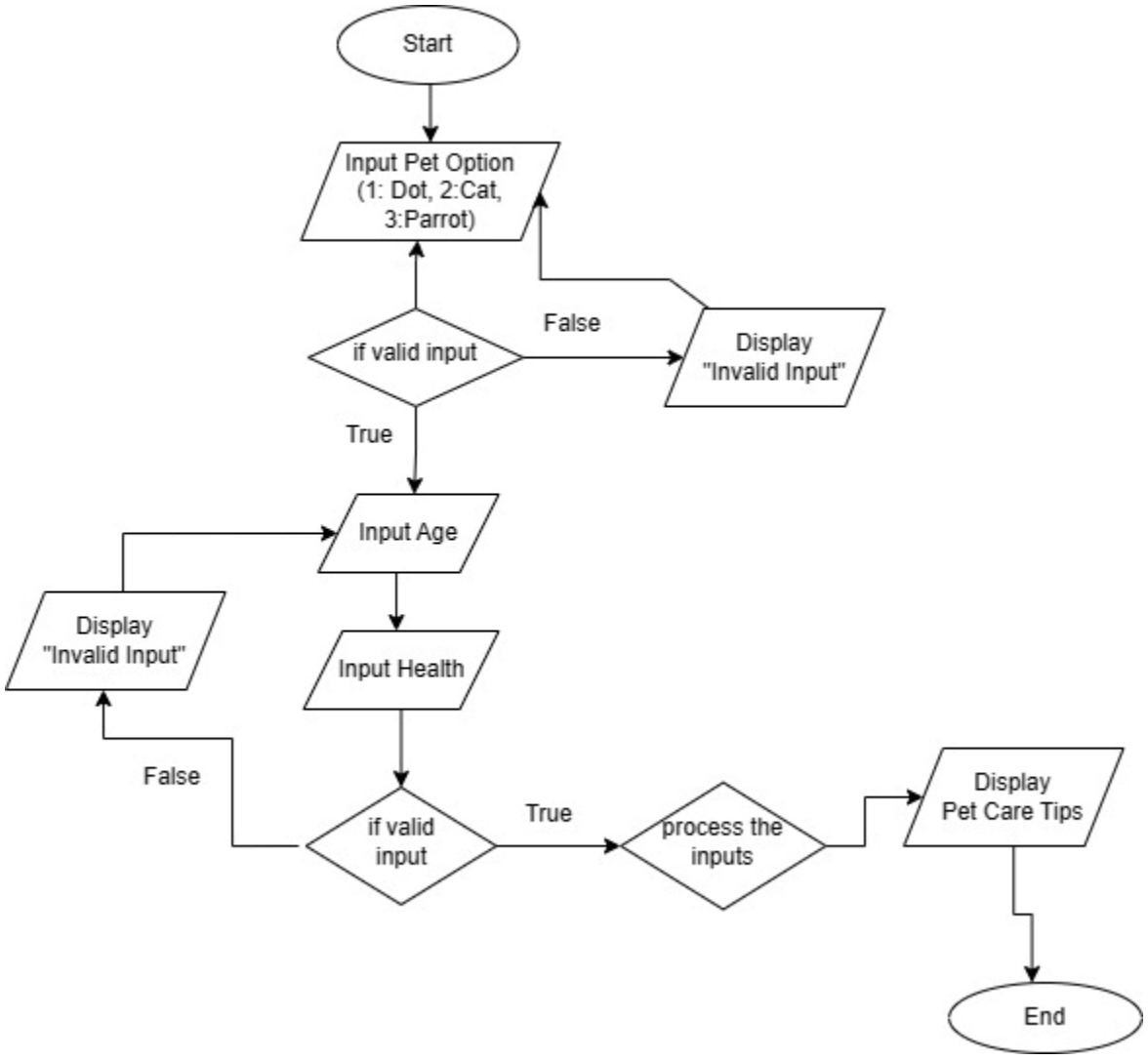
3. **get_pet_health()**: This function prompts the user to select their pet's health status (Good, Fair, Poor) by entering a corresponding number. It ensures the user's input is valid and then returns the health status as a string.

4. **provide_pet_care_tips()**: This function provides pet care tips based on the pet's type (Dog, Cat, Parrot), age, and health status.

5. **pet_care_assistant()**: This is the main function that serves as the entry point for the pet care assistant program. It calls the previous three functions (**get_pet_choice()**, **get_pet_age()**, and **get_pet_health()**) to collect necessary information from the user. Then, it calls the **provide_pet_care_tips()** function to provide personalized care tips based on the user's inputs.

6. **__main__ block**: The program starts by calling the **pet_care_assistant()** function.

Flowchart


Start
Input Pet Option (1: Dot, 2:Cat, 3:Parrot)
if valid input
False
Display "Invalid Input"
True
Input Age
Display "Invalid Input"
Input Health
False
if valid input
True
process the inputs
Display Pet Care Tips
End

Code

Experiment 1

Problem Statement:

Develop a rule-based pet care assistant in Python with the purpose of providing users with helpful reminders and suggestions for pet care. The program will offer personalized care tips based on the type of pet, including dogs, cats, and birds, using predefined rules.

Code

Some Rules to get you started:

Example of a rule for dogs: If the dog is a puppy (i.e. less than 2 years old) frequent training may be needed.

Example of a rule for cats: If the cat's health is poor one may needed onitor their cat closely and consult a vet for any health issues.

Function to get the user's pet choice

```python
# Function to get the user's pet choice
def get_pet_choice():
    """
    TODO: Implement this function to display the pet options and return the user's choice.
    The function should prompt the user to select the type of pet (e.g., dog, cat, bird) and
    return the corresponding number representing the pet type.
    """

    # Display the pet options
    print("1. Dog")
    print("2. Cat")
    print("3. Parrot")

    # Get the user's choice
    while True:
        try:
            pet_choice = int(
                input("Please select the number corresponding to the type of pet you have: "))
            if pet_choice in [1, 2, 3]:
                break
            else:
                print("Invalid choice. Please select a number between 1 and 3.")
        except ValueError:
            print("Invalid input. Please enter a number between 1 and 3.")

    # Return the user's choice
    return pet_choice
```

[1] Python

Function to get pet's age and health as input from user

```python

# Function to get the pet's age from the user
def get_pet_age():
    """
    TODO: Implement this function to get the pet's age from the user.
    The function should prompt the user to enter the pet's age in years and
    return the entered value as an integer.
    """
    # Get the pet's age
    while True:
        try:
            age = float(input("Please enter your pet's age in years: "))
            if age >= 0:
                break
            else:
                print("Age cannot be negative. Please enter a valid age.")
        except ValueError:
            print("Invalid input. Please enter a valid age in years.")

    # Return the pet's age
    return age
```

[2] Python

```python
# Function to get the pet's health status from the user
def get_pet_health():
    """
    TODO: Implement this function to get the pet's health status from the user.
    The function should prompt the user to select the health status (e.g., good, fair, poor) and
    return the corresponding number representing the health status.
    """
    # Get the pet's health status
    print("\nPlease indicate your pet's health status:")
    print("1. Excellent")
    print("2. Good")
    print("3. Fair")
    print("4. Poor")

    while True:
        try:
            health_choice = int(
                input("Select a number corresponding to your pet's health: "))
            if health_choice in [1, 2, 3, 4]:
                if health_choice == 1:
                    health_status = "Excellent"
                elif health_choice == 2:
                    health_status = "Good"
                elif health_choice == 3:
                    health_status = "Fair"
                else:
                    health_status = "Poor"
                break
            else:
                print("Invalid choice. Please select a number between 1 and 4.")
        except ValueError:
            print("Invalid input. Please enter a number between 1 and 4.")

    # Return the pet's health status
    return health_status
```

[3] Python

Function for rules for providing pet care tips

```python
# Function to provide pet care tips based on the user's choice, age, and health
def provide_pet_care_tips(pet_choice, pet_age, pet_health):
    """
    TODO: Implement this function to provide pet care tips based on the user's choices.
    The function should take the pet_choice (representing the type of pet), pet_age (in years),
    and pet_health (representing the health status) as inputs and print relevant care tips based on those.
    """

    if pet_choice == 1:  # Dog
        print("\nTips for your Dog:")
        if pet_age < 2:
            print(
                "- Your dog is a puppy. Frequent training and socialization is recommended.")
        if pet_health == "Poor":
            print("- Monitor your dog closely and consult a vet for any health issues.")
        elif pet_health == "Fair":
            print("- Ensure regular exercise and a balanced diet for your dog.")
        else:
            print("- Continue with regular check-ups and a healthy diet.")

```

```python
    elif pet_choice == 2:  # Cat
        print("\nTips for your Cat:")
        if pet_age < 2:
            print(
                "- Your cat is young. Regular play sessions are beneficial for bonding.")
        if pet_health == "Poor":
            print("- Monitor your cat closely and consult a vet for any health issues.")
        elif pet_health == "Fair":
            print("- Ensure a safe environment and regular check-ups for your cat.")
        else:
            print("- Cats love climbing. Consider getting a cat tree or some shelves.")

    elif pet_choice == 3:  # Parrot
        print("\nTips for your Parrot:")
        if pet_age < 2:
            print("- Your parrot is young. Regular interaction is important for bonding.")
        if pet_health == "Poor":
            print("- Monitor your parrot's behavior and consult a vet if needed.")
        elif pet_health == "Fair":
            print("- Ensure a varied diet and consider toys for mental stimulation.")
        else:
            print("- Continue with regular interaction and check-ups.")
```

[4] Python

Main function to run the pet care assistant

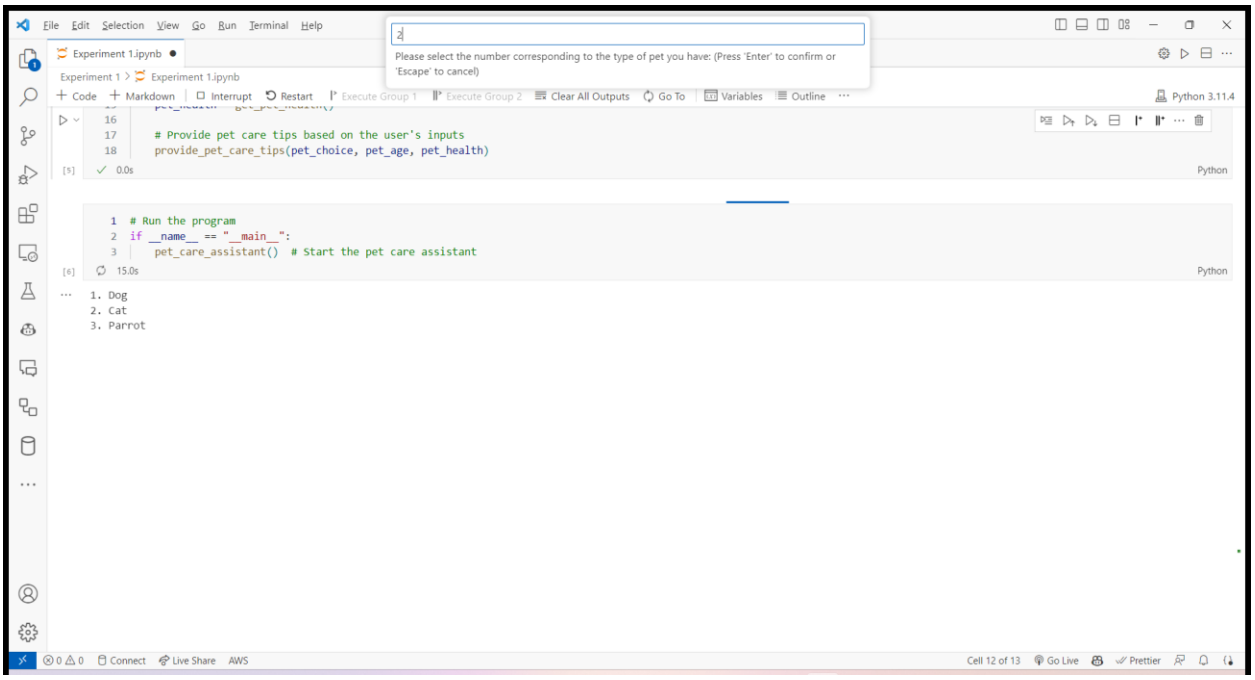
```python
# Main function to run the pet care assistant
def pet_care_assistant():
    """
    TODO: Implement this function as the entry point to the pet care assistant program.
    The function should call the get_pet_choice(), get_pet_age(), and get_pet_health() functions
    to collect the necessary information from the user. Then, it should call the provide_pet_care_tips()
    function to provide personalized care tips based on the user's inputs.
    """

    # Get pet choice from the user
    pet_choice = get_pet_choice()

    # Get pet details (age and health status) from the user
    pet_age = get_pet_age()
    pet_health = get_pet_health()

    # Provide pet care tips based on the user's inputs
    provide_pet_care_tips(pet_choice, pet_age, pet_health)
```

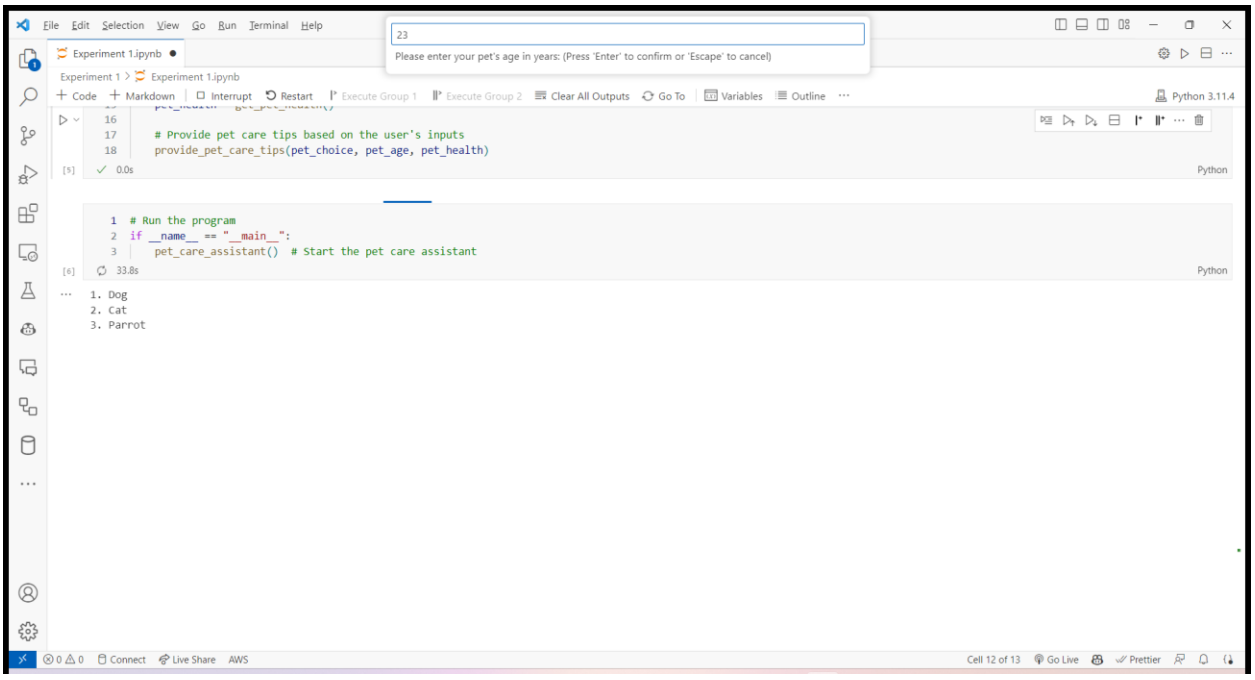
[5] Python

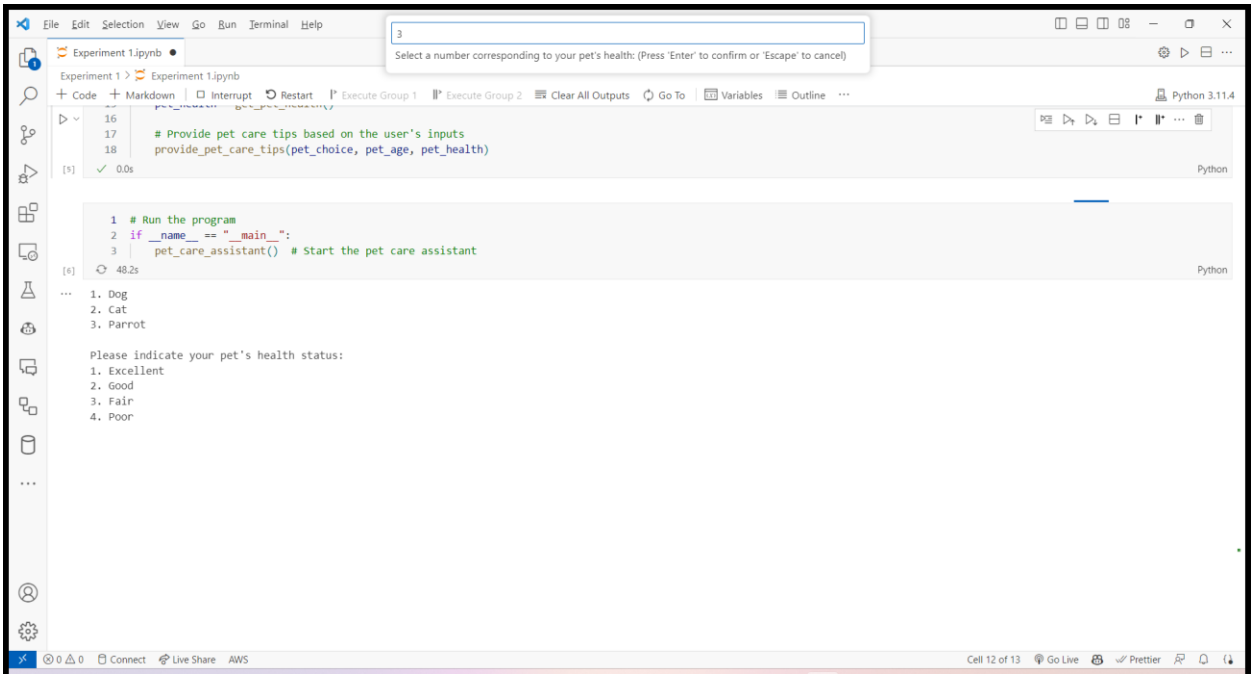
```python
# Run the program
if __name__ == "__main__":
    pet_care_assistant()  # Start the pet care assistant
```

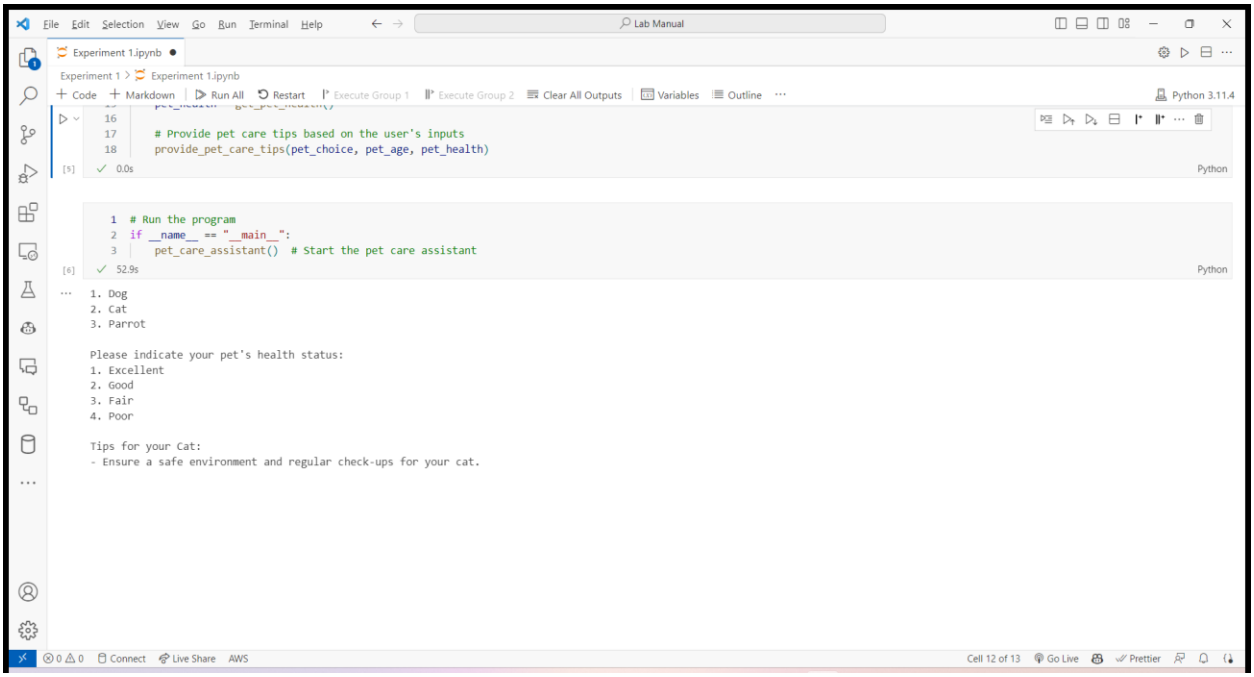
[6] Python

Sample Output






File Edit Selection View Go Run Terminal Help
3
Select a number corresponding to your pet's health: (Press 'Enter' to confirm or 'Escape' to cancel)
Experiment 1.ipynb
Experiment 1 > Experiment 1.ipynb
+ Code + Markdown Interrupt Restart Execute Group 1 Execute Group 2 Clear All Outputs Go To Variables Outline
Python 3.11.4
16
17 # Provide pet care tips based on the user's inputs
18 provide_pet_care_tips(pet_choice, pet_age, pet_health)
[5] 0.0s Python
1 # Run the program
2 if __name__ == "__main__":
3 pet_care_assistant() # Start the pet care assistant
[6] 48.2s Python
1. Dog
2. Cat
3. Parrot
Please indicate your pet's health status:
1. Excellent
2. Good
3. Fair
4. Poor
Connect Live Share AWS
Cell 12 of 13 Go Live Prettier


File Edit Selection View Go Run Terminal Help
Lab Manual
Experiment 1.ipynb
Experiment 1 > Experiment 1.ipynb
+ Code + Markdown Run All Restart Execute Group 1 Execute Group 2 Clear All Outputs Variables Outline
Python 3.11.4
16
17 # Provide pet care tips based on the user's inputs
18 provide_pet_care_tips(pet_choice, pet_age, pet_health)
[5] 0.0s Python
1 # Run the program
2 if __name__ == "__main__":
3 pet_care_assistant() # Start the pet care assistant
[6] 52.9s Python
1. Dog
2. Cat
3. Parrot
Please indicate your pet's health status:
1. Excellent
2. Good
3. Fair
4. Poor
Tips for your Cat:
- Ensure a safe environment and regular check-ups for your cat.
Connect Live Share AWS
Cell 12 of 13 Go Live Prettier

EXPERIMENT NO. 2

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: 19.08.2023
Faculty Signature:
Grade:

Objective(s):

- Understand what breadth first Search (BFS) and Depth first Search (DFS) is.
- Study about different uninformed searching approaches.
- Implement BFS and DFS for solving a real-world problem.

Outcome:

Students would be familiarized with BFS and DFS.
Students would be able to make a comparison between the two algorithms.

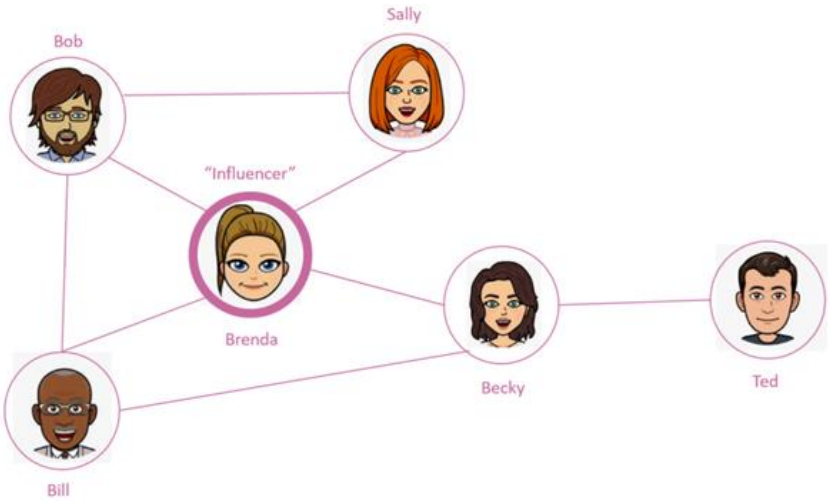
Problem Statement:

Implement Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms in Python to analyze a simple social network. The program aims to explore social connections between users and offer insights into their relationships.

The program should be able to:

- Find groups of users who are directly or indirectly connected to the influencer "*Brenda*".
- Determine if there is a path between two users "*Sally"* and "*Ted"*

Use the graph as provided below for the assignment:



Background Study:

Breadth-First Search (BFS) and Depth-First Search (DFS) are graph traversal algorithms. BFS explores a graph level by level, visiting all neighboring nodes before moving deeper, utilizing a queue data structure. On the other hand, DFS explores as deep as possible along each branch before backtracking, using a stack data structure. Both algorithms are fundamental in graph analysis and have various applications in tasks like path finding, cycle detection, and social network analysis.

Question Bank:

1. What do you understand by blind search algorithms?
   Blind search algorithms are methods used in computer science to explore and traverse problem spaces without having any specific information about the structure or characteristics of the space. These algorithms rely solely on the information available during the search process and do not incorporate domain-specific knowledge.

2. What is *Breadth-First Search* and *Depth-First Search*?
   Blind search algorithms are methods used in computer science to explore and traverse problem spaces without having any specific information about the structure or characteristics of the space. These algorithms rely solely on the information available during the search process and do not incorporate domain-specific knowledge.

3. What are the appropriate scenarios for using BFS and DFS algorithms?
   Breadth-First Search (BFS) is suitable for scenarios where finding the shortest path is important, like in navigation systems or puzzle-solving. It guarantees the shortest path but can require more memory due to the need to store the entire level's nodes.

Depth-First Search (DFS) is appropriate for scenarios where memory efficiency is crucial, such as solving mazes or searching through large trees. It doesn't guarantee the shortest path and can get stuck in infinite loops in some cases, but it's memory efficient as it explores one branch deeply before backtracking.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Pseudocode

BFS Pseudocode

```
FUNCTION path_using_bfs(graph, source_node, destination_node):
    Create a queue and ADD source_node to it.
    Create a set called visited and ADD source_node to it.
    Create a dictionary called path and SET path[source_node] to None.

    WHILE queue is NOT empty:
        current_node = REMOVE first element from queue.

        IF current_node is EQUAL to destination_node:
            RETURN the path from source_node to destination_node using the path dictionary.

        FOR EACH neighbor in graph[current_node]:
            IF neighbor is NOT in visited:
                ADD neighbor to visited.
                SET path[neighbor] to current_node.
                ADD neighbor to queue.

    END FOR
    END WHILE

    RETURN None.

END FUNCTION
```

DFS Pseudocode

```
FUNCTION path_using_dfs(graph, source_node, destination_node):
    Create a stack and PUSH source_node onto it.
    Create a set called visited and ADD source_node to it.
    Create a dictionary called path and SET path[source_node] to None.

    WHILE stack is NOT empty:
        current_node = POP the top element from stack.

        IF current_node is EQUAL to destination_node:
            RETURN the path from source_node to destination_node using the path dictionary.

        FOR EACH neighbor in graph[current_node]:
            IF neighbor is NOT in visited:
                ADD neighbor to visited.
                SET path[neighbor] to current_node.
                PUSH neighbor onto stack.

        END FOR
    END WHILE

    RETURN None.

END FUNCTION
```

Code:

```python
# importing required libraries
from collections import deque
```

[33] Python

Defining the Graph Using Adjacency List

```python
# making the graph
graph = {
    "Brenda": ["Sally", "Bob", "Becky", "Bill"],
    "Bob": ["Brenda", "Sally", "Bill"],
    "Becky": ["Brenda", "Ted", "Bill"],
    "Bill": ["Brenda", "Bob", "Becky", ],
    "Sally": ["Brenda", "Bob"],
    "Ted": ["Becky"]
}
```

[34] Python

```python
def extract_path(path_dict, source, destination):
    if destination not in path_dict:
        return None
    path = []
    current = destination
    while current is not None:
        path.append(current)
        current = path_dict[current]
    path.reverse()
    return path
```

[40] Python

Path Using BFS

```python
def path_using_bfs(graph, source_node, destination_node):
    queue = deque()
    queue.append(source_node)
    visited = set()
    visited.add(source_node)
    path = {}
    path[source_node] = None
    while queue:
        current_node = queue.popleft()
        if current_node == destination_node:
            return extract_path(path, source_node, destination_node)
        for neighbor in graph[current_node]:
            if neighbor not in visited:
                visited.add(neighbor)
                path[neighbor] = current_node
                queue.append(neighbor)
    return None
```

[38] Python

Path Using DFS

```python
def path_using_dfs(graph, source_node, destination_node):
    stack = deque()
    stack.append(source_node)
    visited = set()
    visited.add(source_node)
    path = {}
    path[source_node] = None
    while stack:
        current_node = stack.pop()
        if current_node == destination_node:
            return extract_path(path, source_node, destination_node)
        for neighbor in graph[current_node]:
            if neighbor not in visited:
                visited.add(neighbor)
                path[neighbor] = current_node
                stack.append(neighbor)
    return None
```

[36] Python

Output:

```
[39]
# testing the code
print("BFS")
print(path_using_bfs(graph, "Brenda", "Ted"))
print("DFS")
print(path_using_dfs(graph, "Brenda", "Ted"))
```

Python

```
BFS
['Brenda', 'Becky', 'Ted']
DFS
{'Brenda': None, 'Sally': 'Brenda', 'Bob': 'Brenda', 'Becky': 'Brenda', 'Bill': 'Brenda', 'Ted': 'Becky'}
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 3

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: 26.08.2023
Faculty Signature:
Grade:

Objective(s):

- Understand what State Space Search is.
- Study about how state spaces work and how state space search algorithms work.
- Implement State Space Search for solving a real-world problem.

Outcome:

Student will be familiarized with the State Space Search algorithm.

Problem Statement:

Implement a basic state space search program in Python to solve the classic "Missionaries and Cannibals" puzzle. The goal is to implement a simple program that finds a sequence of valid moves to safely transport three missionaries and three cannibals across a river, following specific constraints.

Background Study:

State space search is a problem-solving technique that navigates through a set of possible states to find a solution. It represents the problem as a graph or tree, with each node representing a state and edges as valid transitions. Algorithms like BFS and DFS are used to explore the state space efficiently, finding solutions for puzzles, games, and optimization tasks by minimizing search effort and avoiding revisiting already explored states using queues and sets. In the "Missionaries and

Cannibals" puzzle, state space search helps identify a valid sequence of moves to safely transport individuals while respecting constraints.

Question Bank:

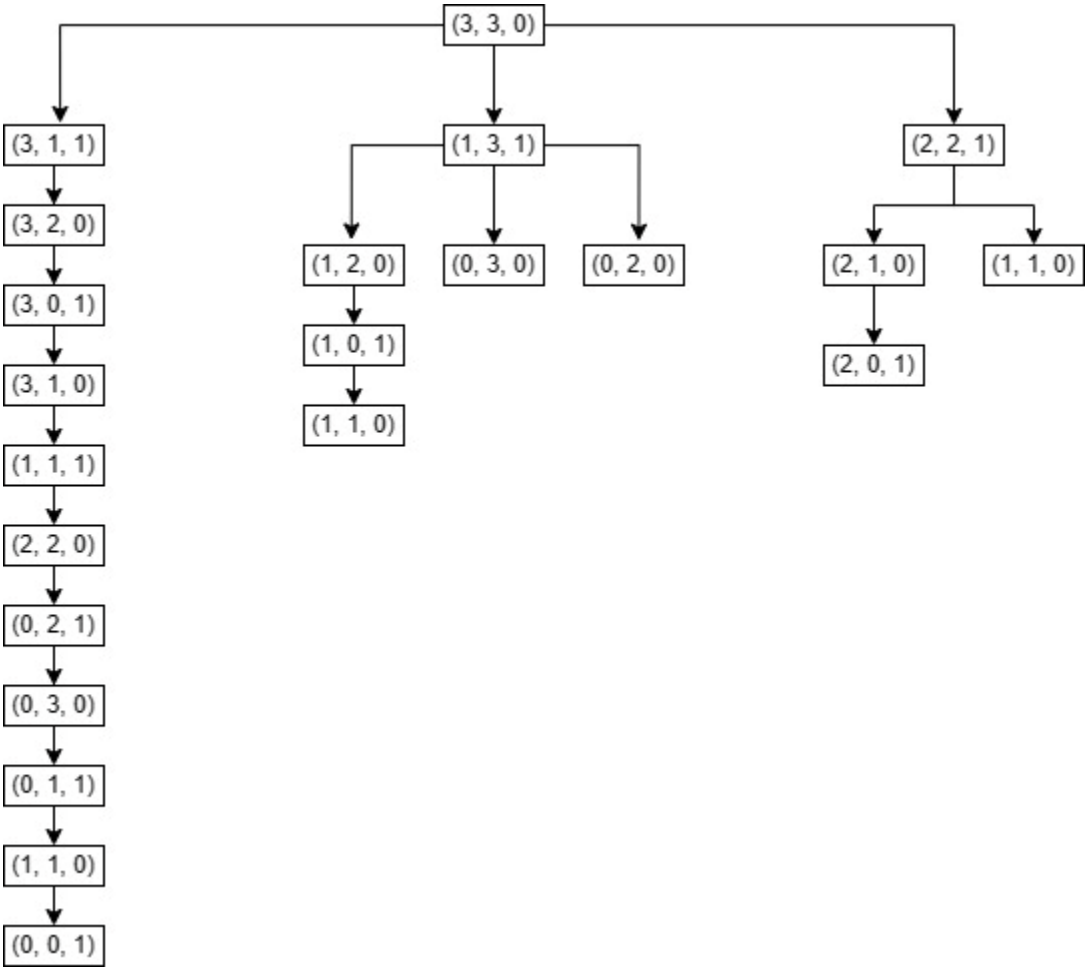
1. What is the state space search technique?
   State space search technique involves systematically exploring the possible states of a problem to find a solution. It's commonly used in AI and computer science to solve problems by representing the problem's possible configurations as states in a graph or tree and then applying search algorithms to traverse and find the optimal solution.

2. Discuss the role of Breadth-First Search (BFS) in solving the "Missionaries and Cannibals" puzzle.
   Breadth-First Search (BFS) plays a crucial role in solving the "Missionaries and Cannibals" puzzle by exploring the state space in a level-by-level manner. In this puzzle, the goal is to move three missionaries and three cannibals from one side of a river to the other using a boat, ensuring that at no point on either side there are more cannibals than missionaries, or the cannibals will eat the missionaries.
   Breadth-First Search (BFS) plays a crucial role in solving the "Missionaries and Cannibals" puzzle by exploring the state space in a level-by-level manner. In this puzzle, the goal is to move three missionaries and three cannibals from one side of a river to the other using a boat, ensuring that at no point on either side there are more cannibals than missionaries, or the cannibals will eat the missionaries.

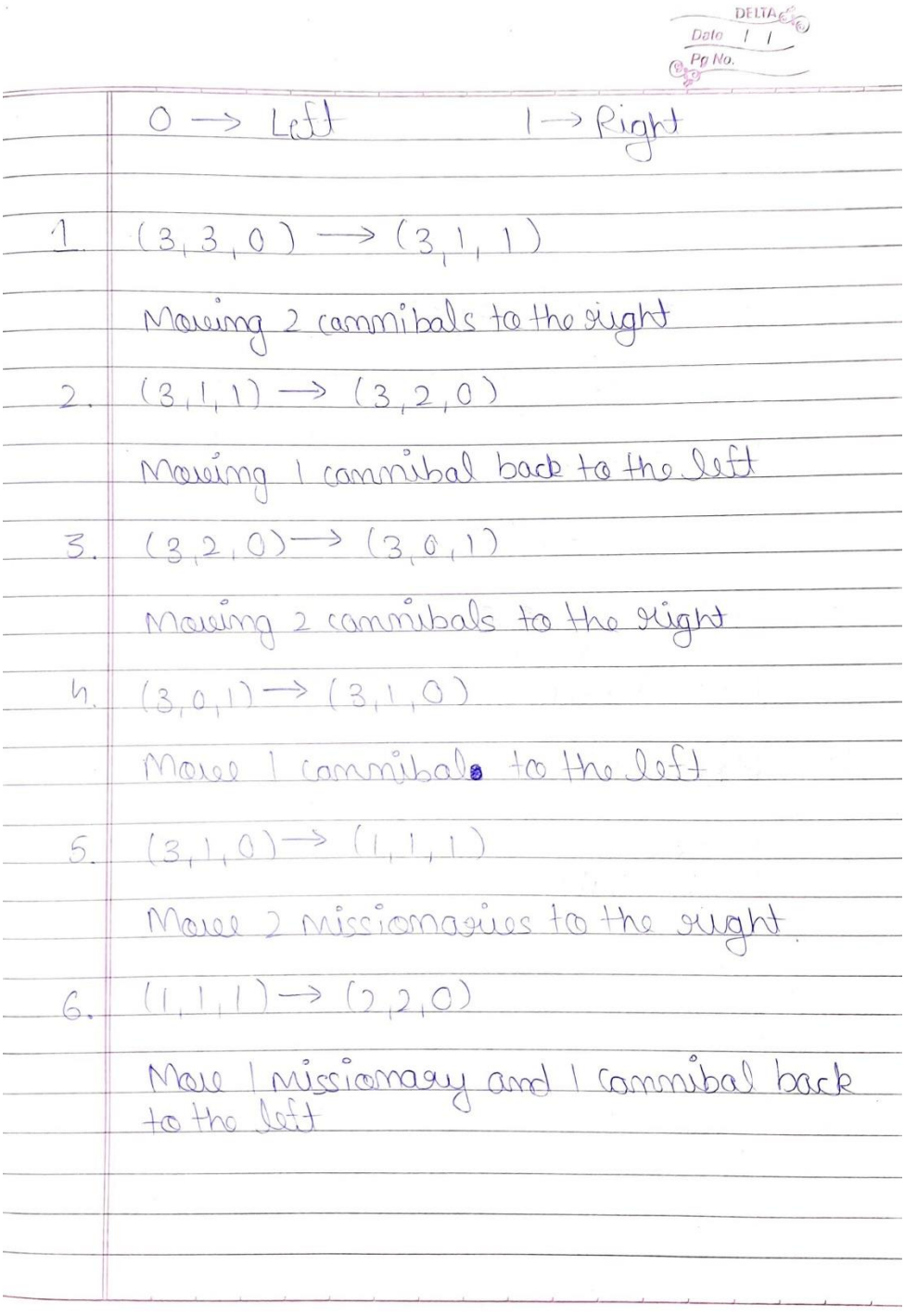
Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

State Space Tree



Manual Solving



0 → Left 1 → Right

1. (3,3,0) → (3,1,1)
   Moving 2 cannibals to the right
2. (3,1,1) → (3,2,0)
   Moving 1 cannibal back to the left
3. (3,2,0) → (3,0,1)
   Moving 2 cannibals to the right
4. (3,0,1) → (3,1,0)
   Move 1 cannibals to the left
5. (3,1,0) → (1,1,1)
   Move 2 missionaries to the right
6. (1,1,1) → (2,2,0)
   Move 1 missionary and 1 cannibal back to the left

DELTA
Date / /
Pg No.

7. (2,2,0) → (0,2,1)
Move 2 missionaries to the right.

8. (0,2,1) → (0,3,0)
Move 1 cannibal back to the left.

9. (0,3,0) → (0,1,1)
Move 2 cannibals to the right

10. (0,1,1) → (1,1,0)
Move 1 ~~cannibal~~ missionary back to left.

11. (1,1,0) → (0,0,1)
Move 1 missionary and 1 cannibal to the right.

(0,0,1) is the required state.

Code

Experiment 3

Problem Statement

Implement a basic state space search program in Python to solve the classic "Missionaries and Cannibals" puzzle. The goal is to implement a simple program that finds a sequence of valid moves to safely transport three missionaries and three cannibals across a river, following specific constraints.

Code

Imports Needed

```
[1] ✓ 0.0s
```

```python
from collections import deque
```

Python

Define the goal state

```
[2] ✓ 0.0s
```

```python
GOAL_STATE = (0, 0, 1)
```

Python

Check if a state is valid

```
[3] ✓ 0.0s
```

```python

def is_valid_state(state):
    (ml, cl, b) = state
    mr = 3 - ml
    cr = 3 - cl
    # Ensure no negative counts and no count greater than 3
    if not (0 <= ml <= 3 and 0 <= cl <= 3 and 0 <= mr <= 3 and 0 <= cr <= 3):
        return False
    # Cannibals shouldn't outnumber missionaries on either bank
    if (ml < cl and ml != 0) or (mr < cr and mr != 0):
        return False
    return True
```

Python

Generate possible next states

```python
def generate_next_states(state):
    (ml, cl, boat) = state
    possible_moves = [(2, 0), (1, 0), (1, 1), (0, 1), (0, 2)]

    next_states = []

    for move in possible_moves:
        if boat == 0:
            next_state = (ml - move[0], cl - move[1], 1)
        else:
            next_state = (ml + move[0], cl + move[1], 0)

        if is_valid_state(next_state):
            next_states.append(next_state)

    return next_states
```

[4] ✓ 0.0s Python

Breadth-First Search function

```python
def bfs(initial_state):
    explored = set()
    queue = deque([(initial_state, [])])

    while queue:
        current_state, path = queue.popleft()

        if current_state == GOAL_STATE:
            return path + [current_state]

        explored.add(current_state)

        for next_state in generate_next_states(current_state):
            if next_state not in explored:
                queue.append((next_state, path + [current_state]))

    return None

```

[5] ✓ 0.0s Python

Output:

Solve the puzzle

```
[6]  ✓ 0.0s                                                    Python

1  # Solve the puzzle
2  initial_state = (3, 3, 0)
3  solution_path = bfs(initial_state)
4
5  if solution_path:
6      print("Solution Path:")
7      for state in solution_path:
8          print(state)
9  else:
10     print("No solution found.")
```

```
Solution Path:
(3, 3, 0)
(2, 2, 1)
(3, 2, 0)
(3, 0, 1)
(3, 1, 0)
(1, 1, 1)
(2, 2, 0)
(0, 2, 1)
(0, 3, 0)
(0, 1, 1)
(1, 1, 0)
(0, 0, 1)
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 4

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: 02.09.2023
Faculty Signature:
Grade:

Objective(s):

- Understand what Greedy Algorithm is.
- Study about different algorithm design paradigms.
- Implement greedy for solving a real-world problem

Outcome:

Students will be familiarized with Greedy Algorithm
Students will be able to make a comparison between the two algorithms.

Problem Statement:

Write a program for the following problem:

A company is planning to launch a new product. They have a limited budget to spend on marketing and advertising. They need to decide how to allocate their budget to maximize the number of people who will be aware of their product.

Marketing Channels:

Social Media: Cost - $50, Reach - 1000 people aware of the product.
Email Campaign: Cost - $80, Reach - 1500 people aware of the product.
Influencer Collaboration: Cost - $120, Reach - 2500 people aware of the product.

Budget Constraint: $200

Now, the company wants to allocate their budget to these marketing channels in such a way that they maximize the total number of people aware of their product.

Background Study:

A greedy algorithm is a heuristic-based technique used in problem-solving. It makes locally optimal choices at each step, hoping that those choices will lead to a global optimal solution. The algorithm selects the best option available at the current state without revisiting or undoing its decisions.

Question Bank:

1. How can you solve a problem using a Greedy approach?

   A Greedy approach involves making locally optimal choices at each step to find the global optimal solution. In a greedy algorithm, you make the best choice available at the moment without considering the consequences of that choice on future steps. Greedy algorithms are useful for optimization problems where finding an exact solution might be complex or time-consuming. While a Greedy approach doesn't guarantee the optimal solution for all problems, it can work well for problems where the greedy choice is always the best choice and doesn't lead to incorrect results.

2. What are the advantages and disadvantages of Greedy algorithm.

   Advantages:

   - Greedy algorithms are often easy to understand and implement.
   - They can be efficient and provide quick solutions for some optimization problems.
   - Greedy algorithms are useful when the problem has a greedy property, where making locally optimal choices leads to a globally optimal solution.

   Disadvantages:

   - Greedy algorithms do not guarantee finding the globally optimal solution in all cases. They might lead to suboptimal solutions.
   - The greedy choice that seems best at the moment might not be the best choice for achieving the overall optimal solution.
   - Determining whether a problem can be solved optimally using a greedy approach can be challenging.
   - Greedy algorithms might require additional verification steps to ensure the solution's correctness.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Pseudocode:

```
function reach_to_cost_ratio_heuristic(channel):
    reach, cost, _ = channel
    return reach / cost

function reach_heuristic(channel):
    reach, _, _ = channel
    return reach

function create_priority_queue(channels, heuristic_function):
    priority_queue = []
    for channel in channels:
        heuristic_value = -heuristic_function(channel)
        priority_queue.append((heuristic_value, channel))
    heapq.heapify(priority_queue)
    return priority_queue
```

```
function greedy_allocation(priority_queue, budget):
    allocated_channels = []
    remaining_budget = budget
    total_cost_used = 0
    total_reach = 0
    while priority_queue and remaining_budget > 0:
        _, channel = heapq.heappop(priority_queue)
        reach, cost, channel_name = channel
        if priority_queue and cost <= remaining_budget:
            allocated_channels.append((channel_name, cost))
            remaining_budget -= cost
            total_cost_used += cost
            total_reach += reach
    return allocated_channels, total_cost_used, total_reach

function main():
    social_media = (1000, 50, "Social Media")
    email_campaign = (1500, 80, "Email Campaign")
    influencer_collaboration = (2500, 120, "Influencer Collaboration")
    marketing_channels = [social_media, email_campaign, influencer_collaboration]
    budget_constraint = 200

    pq_reach_heuristic = create_priority_queue(marketing_channels, reach_heuristic)

    for channel in pq_reach_heuristic:
        print(channel)

    result_reach_heuristic, total_cost_reach_heuristic, total_reach_reach_heuristic = greedy_allocation(pq_reach_heuristic, budget_constraint)

    for channel_name, cost in result_reach_heuristic:
        print(f"{channel_name}: ${cost}")
    print("Total Cost Using Reach Heuristic: $", total_cost_reach_heuristic)
    print("Total Reach Using Reach Heuristic:", total_reach_reach_heuristic)

    allocation_df = create_dataframe(marketing_channels)

    pq_cost_ratio_heuristic = create_priority_queue(marketing_channels, reach_to_cost_ratio_heuristic)

    for channel in pq_cost_ratio_heuristic:
        print(channel)

    result_cost_ratio_heuristic, total_cost_cost_ratio_heuristic, total_reach_cost_ratio_heuristic = greedy_allocation(pq_cost_ratio_heuristic,
budget_constraint)
```

```
    for channel_name, cost in result_cost_ratio_heuristic:
        print(f"{channel_name}: ${cost}")
    print("Total Cost Using Cost-Ratio Heuristic: $", total_cost_cost_ratio_heuristic)
    print("Total Reach Using Cost-Ratio Heuristic:", total_reach_cost_ratio_heuristic)

function create_dataframe(marketing_channels):
    allocation_df = pd.DataFrame(
        {"Channel Name": [channel[2] for channel in marketing_channels],
         "Cost Ratio": [reach_to_cost_ratio_heuristic(channel) for channel in marketing_channels]}
    )
    return allocation_df

if __name__ == "__main__":
    main()
```

Code:

Experiment 4

Problem Statement

Write a program for the following problem: A company is planning to launch a new product. They have a limited budget to spend on marketing and advertising. They need to decide how to allocate their budget to maximize the number of people who will be aware of their product.

Marketing Channels:
Social Media: Cost - $50, Reach - 1000 people aware of the product.
Email Campaign: Cost - $80, Reach - 1500 people aware of the product.
Influencer Collaboration: Cost - $120, Reach - 2500 people aware of the product.

Budget Constraint: $200

Now, the company wants to allocate their budget to these marketing channels in such a way that they maximize the total number of people aware of their product.

The lab report should contain the following items:

1. Priority queue for the problem with reach to cost ratio and only reach as heuristic functions.
2. Pseudo code of the greedy algorithm function for solving the problem
3. Code and output snippets of the assignment.

Code:

```
[327]
# importing requured libraries
import heapq
import pandas as pd
```

Python

Definition of Heuristic Functions

```
[328]
# Define heuristic functions
def reach_to_cost_ratio_heuristic(channel):
    reach, cost, _ = channel
    return reach / cost


def reach_heuristic(channel):
    reach, _, _ = channel
    return reach
```

Python

Priority Queue Creation

```python

def create_priority_queue(channels, heuristic_function):
    """
    Create a priority queue based on a specified heuristic function.

    Args:
        channels (list): List of marketing channels as tuples (reach, cost, channel_name).
        heuristic_function (function): A function to calculate the priority score for a channel.

    Returns:
        list: A priority queue of channels.
    """
    priority_queue = []

    for channel in channels:
        # using negative heuristic value since heapq is a min-heap, but we want max-heap behavior.
        heuristic_value = -heuristic_function(channel)
        priority_queue.append((heuristic_value, channel))

    # making the priority queue
    heapq.heapify(priority_queue)

    # return the priority queue
    return priority_queue
```

[329] Python

Defining the Greedy algorithm

```python
def greedy_allocation(priority_queue, budget):
    """
    Allocate budget greedily based on the priority queue.

    Args:
        priority_queue (list): A priority queue of channels.
        budget (int): The budget constraint.

    Returns:
        list: A list of allocated channels.
    """
    allocated_channels = []
    remaining_budget = budget
    total_cost_used = 0
    total_reach = 0

    while priority_queue and remaining_budget > 0:
        _, channel = heapq.heappop(priority_queue)
        reach, cost, channel_name = channel

        if priority_queue and cost <= remaining_budget:
            allocated_channels.append((channel_name, cost))
            remaining_budget -= cost
            total_cost_used += cost
            total_reach += reach

    return allocated_channels, total_cost_used, total_reach
```

[330] Python

Output:

Main function to solve the problem

```python
# defining the marketing channels as tuples (reach, cost, channel_name)
social_media = (1000, 50, "Social Media")
email_campaign = (1500, 80, "Email Campaign")
influencer_collaboration = (2500, 120, "Influencer Collaboration")

# creating a list of marketing channels
marketing_channels = [social_media, email_campaign, influencer_collaboration]

# maximum budget constraint
budget_constraint = 200


def main():
    """
    Driver function for the marketing budget problem.
    """
    pq_reach_heuristic = create_priority_queue(
        marketing_channels, reach_heuristic)

    print("Priority Queue For Reach Heuristic:")
    for channel in pq_reach_heuristic:
        print(channel)

    result_reach_heuristic, total_cost_reach_heuristic, total_reach_reach_heuristic = greedy_allocation(
        pq_reach_heuristic, budget_constraint)

    print("\nAllocated Channels Using Reach Heuristic:")
    for channel_name, cost in result_reach_heuristic:
        print(f"{channel_name} : ${cost}")
```

```python

    print("Total Cost Using Reach Heuristic: $", total_cost_reach_heuristic)
    print("Total Reach Using Reach Heuristic:", total_reach_reach_heuristic)

    allocation_df = pd.DataFrame(
        {"Channel Name": [channel[2] for channel in marketing_channels],
         "Cost Ratio": [reach_to_cost_ratio_heuristic(channel) for channel in marketing_channels]}
    )
    print("\nCost Ratio For Each Channel:")
    print(allocation_df.to_markdown(index=False))

    pq_cost_ratio_heuristic = create_priority_queue(
        marketing_channels, reach_to_cost_ratio_heuristic)

    print("\nPriority Queue For Cost-Ratio Heuristic:")
    for channel in pq_cost_ratio_heuristic:
        print(channel)

    result_cost_ratio_heuristic, total_cost_cost_ratio_heuristic, total_reach_cost_ratio_heuristic = greedy_allocation(
        pq_cost_ratio_heuristic, budget_constraint)

    print("\nAllocated Channels Using Cost-Ratio Heuristic:")
    for channel_name, cost in result_cost_ratio_heuristic:
        print(f"{channel_name} : ${cost}")

    print("Total Cost Using Cost-Ratio Heuristic: $",
          total_cost_cost_ratio_heuristic)
    print("Total Reach Using Cost-Ratio Heuristic:",
          total_reach_cost_ratio_heuristic)


if __name__ == "__main__":
    main()
```

```
Priority Queue For Reach Heuristic:
(-2500, (2500, 120, 'Influencer Collaboration'))
(-1500, (1500, 80, 'Email Campaign'))
(-1000, (1000, 50, 'Social Media'))

Allocated Channels Using Reach Heuristic:
Influencer Collaboration: $120
Email Campaign: $80
Total Cost Using Reach Heuristic: $ 200
Total Reach Using Reach Heuristic: 4000

Cost Ratio For Each Channel:
| Channel Name              |   Cost Ratio |
|:--------------------------|-------------:|
| Social Media              |      20      |
| Email Campaign            |      18.75   |
| Influencer Collaboration  |      20.8333 |

Priority Queue For Cost-Ratio Heuristic:
(-20.833333333333332, (2500, 120, 'Influencer Collaboration'))
(-18.75, (1500, 80, 'Email Campaign'))
(-20.0, (1000, 50, 'Social Media'))

Allocated Channels Using Cost-Ratio Heuristic:
Influencer Collaboration: $120
Social Media: $50
Total Cost Using Cost-Ratio Heuristic: $ 170
Total Reach Using Cost-Ratio Heuristic: 3500
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 5

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: 09.09.2023
Faculty Signature:
Grade:

Objective(s):

- Understand what Branch and Bound is.
- Study about different algorithm design paradigms.
- Implement Branch and Bound for solving a real-world graph-based problem.

Outcome:

Students will be familiarized with application of Branch and Bound Algorithm on graph-based problems.

Problem Statement:

Write a Python program to solve the Travelling Salesman problem using Branch and Bound approach.

Background Study:

Branch and Bound is a systematic algorithmic technique used to solve optimization problems by exploring the solution space efficiently. It partitions the search space into smaller subproblems, or branches, and establishes upper and lower bounds on their potential solutions. By pruning branches with solutions that cannot possibly improve the current best-known solution, it reduces the search space, leading to faster convergence towards the optimal solution

Question Bank:

1. What is Branch and Bound Approach?
   The Branch and Bound approach is an optimization technique used to solve combinatorial optimization problems. It systematically explores the solution space by dividing it into smaller subproblems or "branches" and then prunes certain branches based on upper and lower bounds. This method is commonly applied to problems like the Traveling Salesman Problem and knapsack problems, helping to find the optimal or near-optimal solutions efficiently.

2. What type of problem is travelling salesman?

The Traveling Salesman Problem is a classic optimization problem in which a salesperson aims to find the shortest possible route that visits a set of cities exactly once and returns to the starting city. It's a well-known NP-hard problem, meaning that as the number of cities increases, finding the exact optimal solution becomes exponentially more difficult. The TSP has applications in various fields such as logistics, transportation, and manufacturing.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Code:

Experiment 5

Problem Statement

Write a Python program to solve the Travelling Salesman problem using Branch and Bound approach.

Imagine a salesman who needs to visit a set of cities and return to his starting point while minimizing the total distance traveled. Let's consider a small set of cities with their pairwise distances:

- City A to City B: 10 miles
- City A to City C: 15 miles
- City A to City D: 20 miles
- City B to City C: 35 miles
- City B to City D: 25 miles
- City C to City D: 30 miles

The goal of the TSP is to find the shortest possible route that visits each city exactly once and returns to the starting city.

Expectation From The Code

1. Cost Matrix
2. Reduced cost matrix
3. All the intermediate matrices (reduced cost) formed during the process to find cost of a path
4. And finally the cost

Code:

```python
import math

# Global Variables
infinity = float('inf')  # Represents an unbounded upper value for comparison
num_nodes = 4  # Total number of nodes in the graph

# Variables to store the result
final_path = [None] * (num_nodes + 1)
final_min_cost = infinity


# Function to update the final_path array
def updateFinalPath(curr_path):
    global num_nodes, final_path
    final_path[:num_nodes + 1] = curr_path[:]
    final_path[num_nodes] = curr_path[0]


# Function to find the minimum edge cost from a given node
def getFirstMinCost(adj_matrix, index):
    return min(adj_matrix[index][j] for j in range(num_nodes) if index != j)


```

```python
# Function to find the second minimum edge cost from a given node
def getSecondMinCost(adj_matrix, index):
    vals = [adj_matrix[index][j] for j in range(num_nodes) if index != j]
    first, second = sorted(vals)[:2]
    return second


# Recursive function to solve the TSP problem
def TSPRecursive(adj_matrix, curr_bound, curr_cost, level, curr_path, visited_nodes):
    global final_min_cost, num_nodes

    # base case: if we have reached the last node and there is an edge
    # from the last node to the first node
    if level == num_nodes:
        if adj_matrix[curr_path[level - 1]][curr_path[0]] != 0:
            curr_total_cost = curr_cost + \
                adj_matrix[curr_path[level - 1]][curr_path[0]]
            if curr_total_cost < final_min_cost:
                updateFinalPath(curr_path)
                final_min_cost = curr_total_cost
        return

    # Loop through all vertices and recurse
    for i in range(num_nodes):
        if adj_matrix[curr_path[level - 1]][i] != 0 and visited_nodes[i] == False:
            temp_bound = curr_bound
            curr_cost += adj_matrix[curr_path[level - 1]][i]

            # Calculate a new lower bound
            if level == 1:
                curr_bound -= ((getFirstMinCost(adj_matrix, curr_path[level - 1]) +
                                getFirstMinCost(adj_matrix, i)) / 2)
            else:
                curr_bound -= ((getSecondMinCost(adj_matrix, curr_path[level - 1]) +
                                getFirstMinCost(adj_matrix, i)) / 2)

            # If the new lower bound + current cost is less than final_min_cost,
            # continue with this path
            if curr_bound + curr_cost < final_min_cost:
                curr_path[level] = i
                visited_nodes[i] = True

                TSPRecursive(adj_matrix, curr_bound, curr_cost,
                             level + 1, curr_path, visited_nodes)

            # Reset variables for next iteration
            curr_cost -= adj_matrix[curr_path[level - 1]][i]
            curr_bound = temp_bound
            visited_nodes[i] = False

```

```python
def TSP(adj_matrix):
    global final_min_cost, num_nodes

    # Initialize variables for TSP
    curr_bound = 0
    curr_path = [-1] * (num_nodes + 1)
    visited_nodes = [False] * num_nodes

    # Calculate initial lower bound
    for i in range(num_nodes):
        curr_bound += (getFirstMinCost(adj_matrix, i) +
                       getSecondMinCost(adj_matrix, i))
    curr_bound = math.ceil(curr_bound / 2)

    # Start from vertex 0
    visited_nodes[0] = True
    curr_path[0] = 0

    # Call recursive TSP function
    TSPRecursive(adj_matrix, curr_bound, 0, 1, curr_path, visited_nodes)

    # Print the final result
    print("\nMinimum cost:", final_min_cost)
    print("Path Taken:", ' '.join(map(str, final_path)))


```

Output:

```python
# Example adjacency matrix
adj_matrix = [[0, 10, 15, 20],
              [10, 0, 35, 25],
              [15, 35, 0, 30],
              [20, 25, 30, 0]]

print("Cost Matrix: ")
for row in adj_matrix:
    print(row)


TSP(adj_matrix)
```

[11] ✓ 0.0s Python

```
Cost Matrix:
[0, 10, 15, 20]
[10, 0, 35, 25]
[15, 35, 0, 30]
[20, 25, 30, 0]

Minimum cost: 80
Path Taken: 0 1 3 2 0
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 6

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: 16.09.2023
Faculty Signature:
Grade:

Objective(s):

- Understand and study about Triples in a knowledge graph.
- Form triples and implement a knowledge graph from it.

Outcome:

Students will be familiarized with Triples creation and knowledge graph implementation from it.

Problem Statement:

Form triples based on the following paragraph:

"Alice is enrolled in Computer Science 101. Bob is enrolled in Physics 201. Charlie is enrolled in Mathematics 301. Computer Science 101 is taught by Professor Smith. Physics 201 is taught by Professor Johnson. Mathematics 301 is taught by Professor Brown."

Use the above to paragraph extract triples and build a complete graph representing the relationships between students, courses, and instructors in a university setting.

Background Study:

Triples creation involves representing data using three components: subject, predicate, and object, which describe relationships between entities. A knowledge graph is a data structure that utilizes triples to organize and store information in a graph format. It enables efficient storage, retrieval, and analysis of complex relationships, making it valuable for gaining insights and answering questions about interconnected data. Knowledge graphs find applications in various domains, such as AI, recommendation systems, and semantic web technologies, as they facilitate comprehensive knowledge representation and inference.

Question Bank:

1. What is the format of Triples?

The format of Triples in the context of knowledge graphs typically follows the RDF (Resource Description Framework) format, which consists of three parts: Subject, Predicate, and Object. These parts are represented as <Subject, Predicate, Object>. For example, <John, hasAge, 30> represents the triple where John has an age of 30.

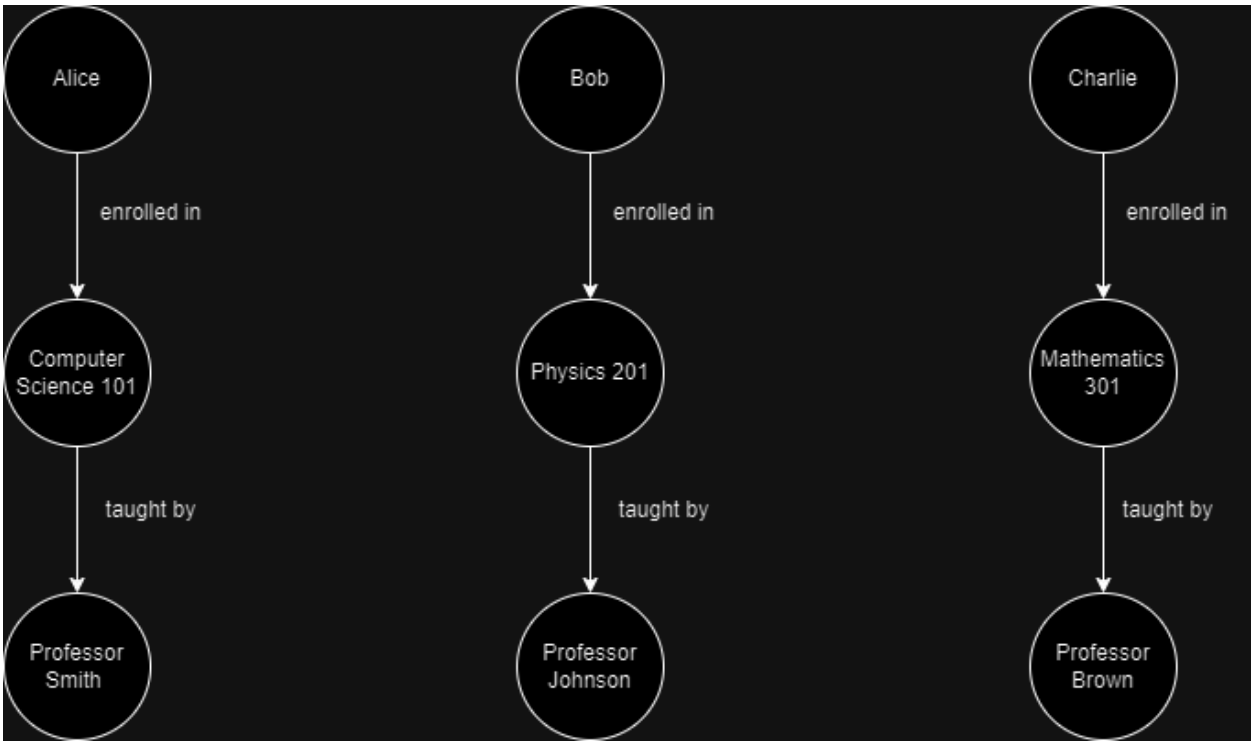
2. Explain how triples help in creating a knowledge graph.

   Triples help in creating a knowledge graph by representing structured information in a way that's easily understandable by both humans and machines. Each triple encodes a specific relationship between a subject and an object using a predicate. When you have many such triples, they can be linked together to form a network of knowledge, where entities and their relationships are interconnected. This graph structure allows for efficient data retrieval, reasoning, and the discovery of new insights from the interconnected information.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Knowledge Graph:



Code:

Experiment 6

Problem Statement

Form triples based on the following paragraph:

"Alice is enrolled in Computer Science 101. Bob is enrolled in Physics 201. Charlie is enrolled in Mathematics 301. Computer Science 101 is taught by Professor Smith. Physics 201 is taught by Professor Johnson. Mathematics 301 is taught by Professor Brown."

Use the above to paragraph extract triples and build a complete graph representing the relationships between students, courses, and instructors in a university setting.

Code:

```python
# To plot a networkx graph in pyvis
import networkx as nx
from pyvis.network import Network
from IPython.display import HTML
from IPython.display import display, IFrame
import matplotlib.pyplot as plt
```

[42] Python

Definition of the Triples

```python
# Manually define the triples from the paragraph in the subject predicate and object format as a list of tuples
triples = [
    ("Alice", "Computer Science 101", "enrolled_in"),
    ("Bob", "Physics 201", "enrolled_in"),
    ("Charlie", "Mathematics 301", "enrolled_in"),
    ("Computer Science 101", "Professor Smith", "taught_by"),
    ("Physics 201", "Professor Johnson", "taught_by"),
    ("Mathematics 301", "Professor Brown", "taught_by")
]
```

[43] Python

Graph Building using Networkx

```python
# Function to build a NetworkX graph from extracted triples
def build_networkx_graph(triples):
    """
    Builds a NetworkX graph from a list of subject-predicate-object triples.

    Args:
        triples (list): A list of extracted triples, each represented as a tuple (subject, predicate, object).

    Returns:
        networkx.Graph: A NetworkX graph representing relationships between students, courses, and instructors.
    """
    # TO-DO: Implement the code to build a NetworkX graph from the triples
    # Initialize an empty NetworkX graph
    G = nx.Graph()
    for subj, obj, pred in triples:
        G.add_edge(subj, obj, label=pred)
    return G
```

[44] Python

Graph Visualize using Pyvis

```python
# Function to save the graph as "university_relationship_graph.html" using PyVis
def save_graph_pyvis(graph):
    """
    Visualizes a NetworkX graph using PyVis and saves it as an HTML file.

    Args:
        graph (networkx.Graph): The NetworkX graph to be visualized.

    Returns:
        None
    """
    # Create an empty PyVis Network object
    net = Network(notebook=True)

    # Add nodes and edges to the PyVis graph
    for node in graph.nodes():
        net.add_node(node, label=node)

    for edge in graph.edges():
        net.add_edge(edge[0], edge[1], label=graph[edge[0]][edge[1]]['label'])

    # Save the graph as an HTML file
    net.show("university_relationship_graph.html")
```

[45] Python

Graph Visualize using Matplotlib

```python
def draw_graph_matplotlib(graph):
    plt.figure(figsize=(15, 10))
    pos = nx.spring_layout(graph, k=1)
    nx.draw(graph, pos, with_labels=True, node_color='gray', node_size=4000, font_size=10)
    edge_labels = {(u, v): graph[u][v]['label'] for u, v in graph.edges()}
    nx.draw_networkx_edge_labels(
        graph, pos, edge_labels=edge_labels, font_color='red', font_size=10)
    plt.show()
```

[46] Python

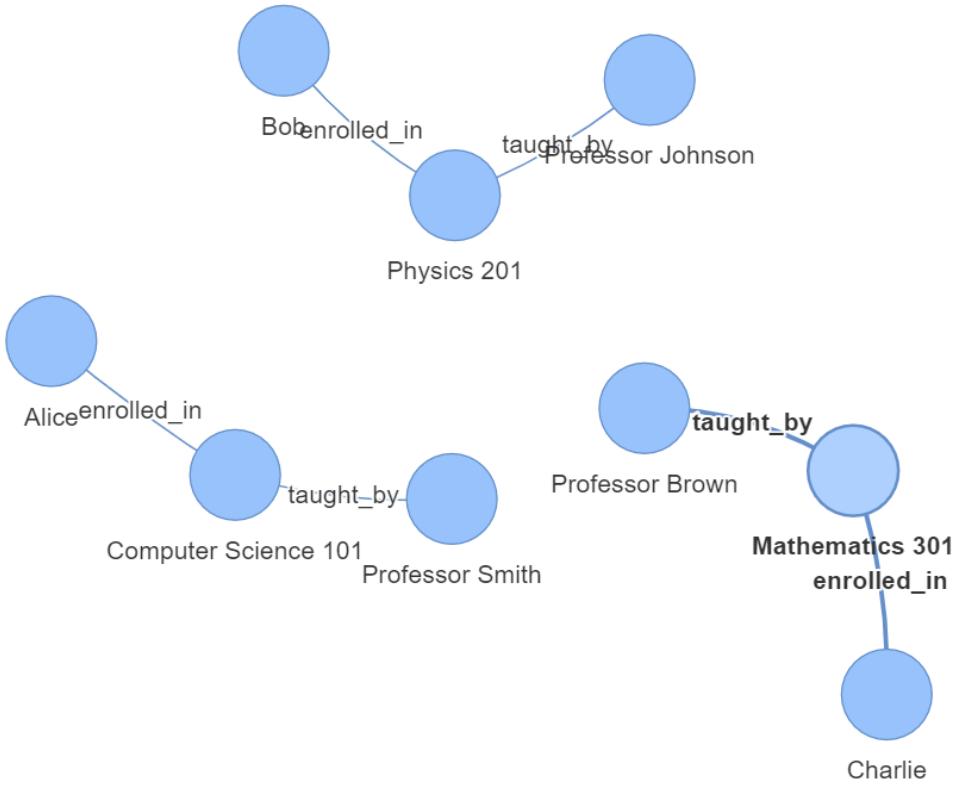
Output:

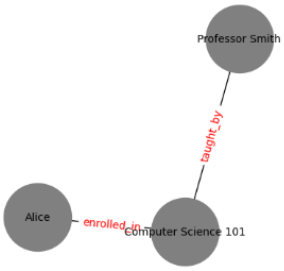
Main function to solve the problem

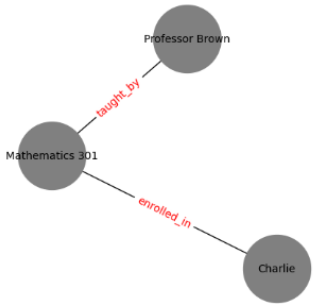
```
# Build a graph using the manually defined triples
graph = build_networkx_graph(triples)

# Save the graph using PyVis
save_graph_pyvis(graph)

draw_graph_matplotlib(graph)
```

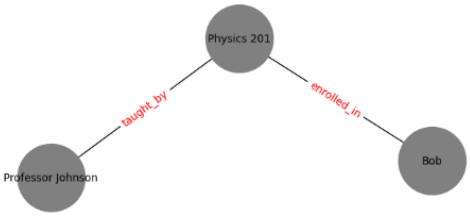
[47] Python

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues displaying graphics on chrome/safari. Use cdn_resources='in_line' or cdn_resources='remote' if you have issues view
university_relationship_graph.html
```




Professor Smith
taught_by
Alice
enrolled_in
Computer Science 101


Professor Brown
taught_by
Mathematics 301
enrolled_in
Charlie


Physics 201
taught_by
enrolled_in
Professor Johnson
Bob

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 7

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:
Faculty Signature:
Grade:

Objective(s):

- Understand and study Bayesian Network.
- Implement Bayesian Network for solving a real-world graph-based problem.

Outcome:

Students will be familiarized with concepts of Bayesian Network.

Problem Statement:

Using Bayesian network, implement a Python program to calculate probability in disease diagnosis.

A medical clinic is trying to determine the probability that a patient has a certain disease. They have a test that can be used to diagnose the disease, but the test is not always accurate. The clinic knows that the probability of a false positive is 1%, and the probability of a false negative is 5%. The clinic has a patient who has tested positive for the disease.
They want to use Bayesian Theorem to calculate the probability that the patient actually has the disease. The Bayesian network will consist of two nodes: 'D' (for disease) and 'T' (for test result). conditional probability distributions (CPDs) for the nodes will be based on the provided information

Background Study:

Bayesian networks, also known as belief networks or probabilistic graphical models, are powerful tools for representing and reasoning about uncertainty in probabilistic systems. They are widely used in various fields, including medical diagnosis, natural language processing, finance, and more. A Bayesian network is a directed acyclic graph (DAG) in which nodes represent random variables, and edges represent probabilistic dependencies between the variables.

Question Bank:

1. What is a Bayesian Theorem?

The Bayesian Theorem, also known as Bayes' Theorem, is a fundamental concept in probability theory and statistics. It describes the probability of an event occurring based on prior knowledge or information. The theorem mathematically relates the conditional probability of an event A given event B, with the conditional probability of event B given event A, along with the probabilities of events A and B independently.

2. Discuss the applications of Bayesian Theorem?
   - Medical Diagnosis: Bayesian methods help in disease diagnosis by incorporating prior probabilities and test results to estimate the probability of a patient having a particular condition.
   - Spam Filtering: Bayesian algorithms are used in spam email filters, considering word probabilities to determine the likelihood of an email being spam.
   - Machine Learning: Bayesian networks assist in modeling dependencies between variables, aiding in probabilistic reasoning and prediction tasks.
   - Natural Language Processing: Bayesian methods are employed in language models, part-of-speech tagging, and sentiment analysis.
   - Recommendation Systems: Bayesian techniques help in collaborative filtering and content-based recommendation to suggest relevant products or content.
   - Financial Analysis: Bayesian inference is used in risk assessment, portfolio optimization, and credit scoring.
   - Weather Forecasting: Bayesian models can integrate historical data and current observations to improve the accuracy of weather predictions.
   - Fault Diagnosis: In engineering, Bayesian networks aid in diagnosing faults in complex systems by analyzing sensor data and known relationships.
   - Image Processing: Bayesian techniques enhance image denoising, object recognition, and image segmentation.

   Bayesian methods are versatile and applicable in various fields where uncertainty and probabilistic reasoning are involved.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 08

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: 23.09.2023
Faculty Signature:
Grade:

Objective(s):

- Understand and study Visualization techniques for graphs.
- Apply logical reasoning over the visualized graph.

Outcome:

Students will be familiarized with Graph Based Visualization and applying logical reasoning over the graph.

Problem Statement:

Implement a Python Code for the following problem:
A logistics company is trying to optimize their delivery routes. They have a dataset of historical delivery data, which includes the start and end points of each delivery, as well as the distance between each point. They want to use graph-based visualization and logical reasoning to identify the most efficient delivery routes.
The dataset is:

Delivery ID	Start Point	End Point	Distance (in miles)
1	Warehouse	Point A	10
2	Point A	Point B	5
3	Point A	Point C	8
4	Point B	Point C	7
5	Point B	Point D	12
6	Point C	Point D	6
7	Point C	Point E	9
8	Point D	Point E	11

Background Study:
Graph-based visualization is a powerful technique for representing and understanding complex relationships and connections among various data elements. It involves creating visual representations of data as nodes (vertices) connected by edges (lines) that indicate the relationships between them. Graphs allow for a clear depiction of patterns, clusters, and dependencies, enabling users to uncover insights that might be less apparent in raw data.

Question Bank:

1. How can you visualize graphs from a given dataset?
   Visualizing Graphs from a Dataset: Graphs can be visualized from a dataset using graph visualization tools or libraries like NetworkX (Python), Gephi, or D3.js. These tools help represent nodes (vertices) and edges, allowing you to visualize relationships and structures present in the data.

2. Which algorithms could have been applied to get identify the efficient delivery routes?
   - Dijkstra's Algorithm: Used for finding the shortest path between nodes in a weighted graph, applicable to identifying efficient routes.
   - A Algorithm*: Combines Dijkstra's with heuristics for optimal pathfinding in graphs, often used in route planning with distance and estimated cost considerations.
   - Traveling Salesman Problem (TSP) Algorithms: Various algorithms exist to solve the TSP, including Genetic Algorithms, Ant Colony Optimization, and Dynamic Programming.
   - Floyd-Warshall Algorithm: Finds shortest paths between all pairs of nodes in a weighted graph, useful for identifying optimal routes in delivery networks.
   - Constrained Shortest Path Algorithms: Incorporates constraints like time windows, capacity, and vehicle availability into route optimization.

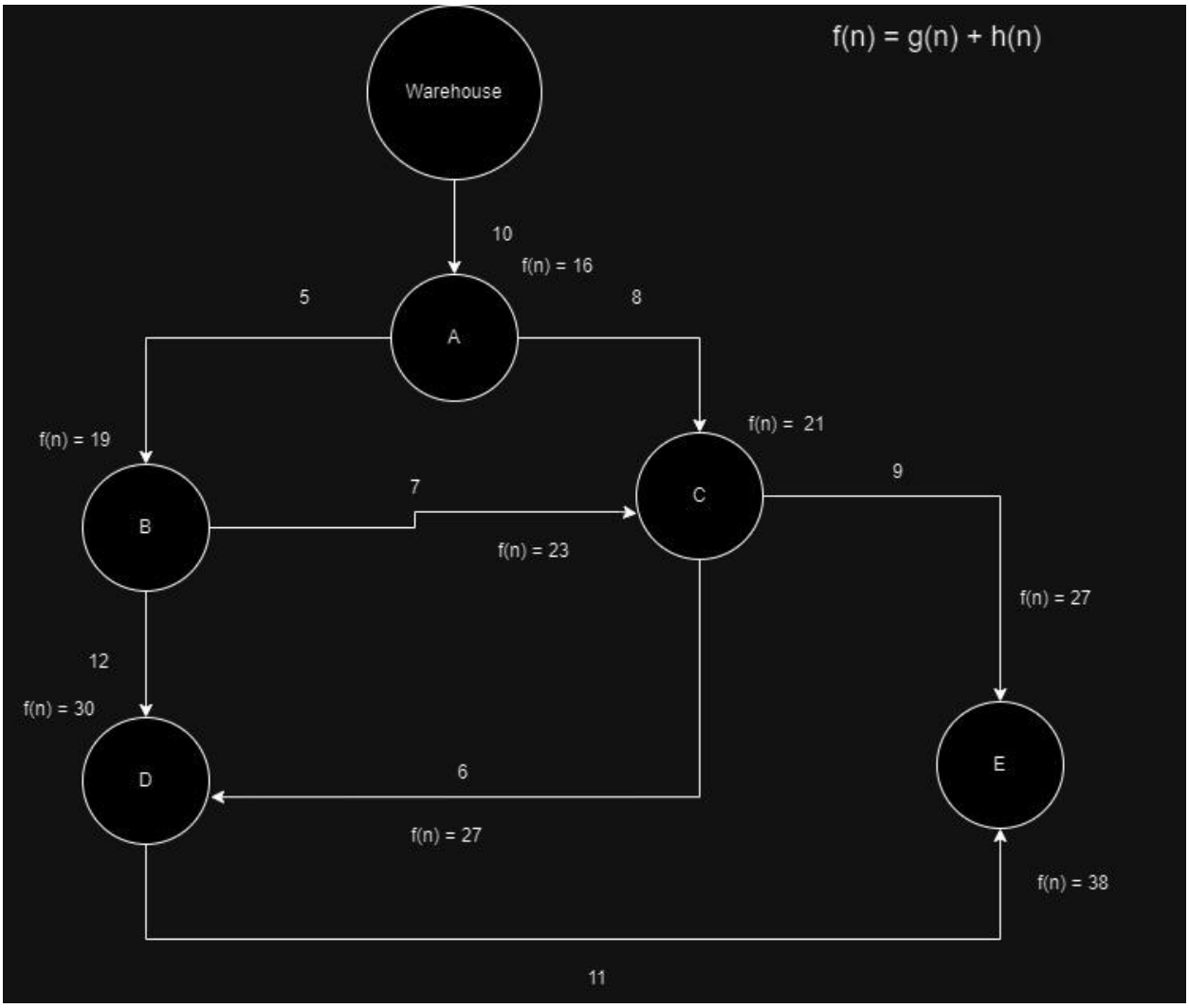
Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Pseudocode

```
Initialize empty priority queue pq with (0, start, empty list, 0)
Initialize empty set visited

While pq is not empty:
    Pop (priority, current, path, cost) from pq
    If current is visited:
        Continue
    Add current to visited
    Update path by appending current

    If current equals end:
        Return path and cost

    For each neighbor of current:
        If current or end not in heuristic_table:
            Continue
        Get weight from graph for edge (current, neighbor)
        Get heuristic from heuristic_table for (current, end)
        Calculate new_cost = cost + weight
        Calculate new_priority = new_cost + heuristic

        Add (new_priority, neighbor, path, new_cost) to pq

If function reaches this point:
    Return None, None
```

Graph


f(n) = g(n) + h(n)
Warehouse
10
f(n) = 16
5
8
A
f(n) = 19
f(n) = 21
B
7
C
9
f(n) = 23
f(n) = 27
12
f(n) = 30
D
6
E
f(n) = 27
f(n) = 38
11

Delivery ID	Start Point	End Point	Distance (in miles)
1	Warehouse	Point A	10
2	Point A	Point B	5
3	Point A	Point C	8
4	Point B	Point C	7
5	Point B	Point D	12
6	Point C	Point D	6
7	Point C	Point E	9
8	Point D	Point E	11

Heuristic table

	A	B	C	D	E
W	6	5	4	5	6
A	5	4	3	4	5
B	4	3	2	3	4
C	3	2	1	2	3
D	4	3	2	3	4
E	5	4	3	4	5

Shortest Logical Path: Warehouse -> Point A -> Point C -> Point E

Code:

Experiment 7

Problem Statement:

Implement a Python Code for the following problem: A logistics company is trying to optimize their delivery routes. They have a dataset of historical delivery data, which includes the start and end points of each delivery, as well as the distance between each point. They want to use graph-based visualization and logical reasoning to identify the most efficient delivery routes.

Delivery ID	Start Point	End Point	Distance (in miles)
1	Warehouse	Point A	10
2	Point A	Point B	5
3	Point A	Point C	8
4	Point B	Point C	7
5	Point B	Point D	12
6	Point C	Point D	6
7	Point C	Point E	9
8	Point D	Point E	11

Heuristic table

	A	B	C	D	E
W	6	5	4	5	6
A	5	4	3	4	5
B	4	3	2	3	4
C	3	2	1	2	3
D	4	3	2	3	4
E	5	4	3	4	5

Code:

```python
# To plot a networkx graph in pyvis
import numpy as np
import pandas as pd
import networkx as nx
from pyvis.network import Network
import heapq
```

[12] Python

Graph Building using Networkx

```python
# Step 1: Create a graph representation of delivery routes
def create_delivery_graph(data):
    # TODO: Implement this function to create a graph from the given delivery data using NetworkX
    G = nx.DiGraph()
    for _, start, end, distance in data:
        G.add_edge(start, end, weight=distance)
        G.add_edge(end, start, weight=distance)
    return G
```

[13] Python

Graph Visualize using Pyvis

+ Code + Markdown

```python
# Step 2: Visualize the graph using Pyvis
def visualize_graph(graph):
    # TODO: Implement this function to visualize the graph using Pyvis
    # Create an empty PyVis Network object
    net = Network(notebook=True)

    # Add nodes and edges to the PyVis graph
    for node in graph.nodes():
        net.add_node(node, label=node)
    for edge in graph.edges():
        weight = graph[edge[0]][edge[1]]['weight']
        net.add_edge(edge[0], edge[1], title=str(weight), label=str(weight))

    # Save the graph as an HTML file
    net.show("delivery_routes_graph.html")
```

[14] Python

A* Search Algorithm

```python
# Step 3: Implement A* heuristic search algorithm


def a_star_search(graph, start, end, heuristic_table):
    pq = [(0, start, [start], 0)]
    visited = set()

    while pq:
        priority, current, path, cost = heapq.heappop(pq)

        if current in visited:
            continue
        visited.add(current)

        if current == end:
            return path, cost

        for neighbor in graph.neighbors(current):
            if neighbor in visited:
                continue

            weight = graph[current][neighbor]['weight']
            heuristic = 0
            if neighbor in heuristic_table.index:
                heuristic = heuristic_table.loc[neighbor, end]

            new_cost = cost + weight
            new_priority = new_cost + heuristic

            new_priority = new_cost + heuristic

            heapq.heappush(pq, (new_priority, neighbor,
                                path + [neighbor], new_cost))

    return None, None
```

[15] Python

Main function to solve the problem

```python
if __name__ == "__main__":
    # Sample dataset
    delivery_data = [
        (1, 'Warehouse', 'Point A', 10),
        (2, 'Point A', 'Point B', 5),
        (3, 'Point A', 'Point C', 8),
        (4, 'Point B', 'Point C', 7),
        (5, 'Point B', 'Point D', 12),
        (6, 'Point C', 'Point D', 6),
        (7, 'Point C', 'Point E', 9),
        (8, 'Point D', 'Point E', 11)
    ]

    # making delivery_data into a pandas dataframe
    delivery_data_df = pd.DataFrame(delivery_data, columns=[
        'id', 'start', 'end', 'distance'])

    # printing the dataframe
    print(delivery_data_df.to_markdown(), end="\n\n")
```

```python

    # Create the heuristic table
    heuristic_table = pd.DataFrame({
        'Warehouse': [6, 5, 4, 5, 6],
        'Point A': [5, 4, 3, 4, 5],
        'Point B': [4, 3, 2, 3, 4],
        'Point C': [3, 2, 1, 2, 3],
        'Point D': [4, 3, 2, 3, 4],
        'Point E': [5, 4, 3, 4, 5]
    }, index=['Point A', 'Point B', 'Point C', 'Point D', 'Point E'])

    print(heuristic_table.to_markdown(), end="\n\n")

    # Create the delivery graph
    delivery_graph = create_delivery_graph(delivery_data)

    # Visualize the graph
    visualize_graph(delivery_graph)

    # Find the shortest distance using A* heuristic search
    start_point = 'Warehouse'
    end_point = 'Point E'
    shortest_path, shortest_distance = a_star_search(
        delivery_graph, start_point, end_point, heuristic_table)

    if shortest_path:
        print(
            f"Shortest path from {start_point} to {end_point}: {' -> '.join(shortest_path)}")
        print(f"Shortest distance: {shortest_distance} miles")
    else:
        print(f"No path found from {start_point} to {end_point}")
```

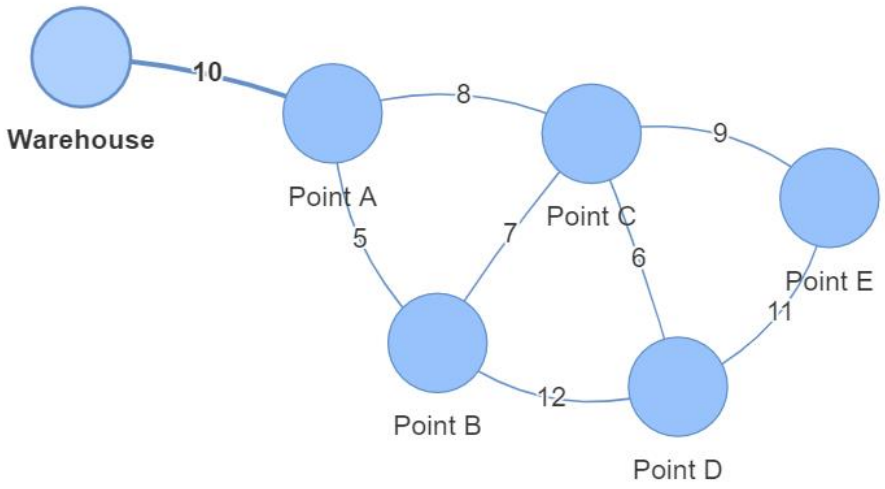
[16] Python

Output:

	id	start	end	distance
0	1	Warehouse	Point A	10
1	2	Point A	Point B	5
2	3	Point A	Point C	8
3	4	Point B	Point C	7
4	5	Point B	Point D	12
5	6	Point C	Point D	6
6	7	Point C	Point E	9
7	8	Point D	Point E	11

	Warehouse	Point A	Point B	Point C	Point D	Point E
Point A	6	5	4	3	4	5
Point B	5	4	3	2	3	4
Point C	4	3	2	1	2	3
Point D	5	4	3	2	3	4
Point E	6	5	4	3	4	5

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues displaying graphics on chrome/safari. Use cdn_resources='in_line' or cdn_resources='remote' if you have issues view
delivery_routes_graph.html
Shortest path from Warehouse to Point E: Warehouse -> Point A -> Point C -> Point E
Shortest distance: 27 miles
```



THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 09

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: 28.10.2023
Faculty Signature:
Grade:

Objective(s):

- Understand and study Simple Expert Systems.

Outcome:

Students will be familiarized with *Simple Expert Systems.*

Problem Statement:

A manufacturing company is trying to improve their quality control process. They have a dataset of historical quality data, which includes the results of quality tests, as well as the specifications for each product. Implement a Simple Expert system to identify potential quality problems.

The dataset is:

Product ID	Test 1 Result	Test 2 Result	Test 3 Result	Specification 1	Specification 2	Specification 3
1	95	20	8	100	25	10
2	98	22	9	100	20	10
3	93	18	7	95	15	8
4	100	24	10	100	25	12
5	96	21	8	98	20	10
6	92	19	6	95	15	8
7	90	17	5	95	15	8

Background Study:

A simple expert system is a type of artificial intelligence that uses a set of predefined rules or knowledge to make decisions or solve problems in a specific domain. It is designed to mimic the decision-making process of a human expert in a particular field. The system consists of a knowledge base, which stores the rules and facts about the domain, and an inference engine, which uses logical reasoning to derive conclusions from the available knowledge.

Question Bank:

1. What are Expert Systems?
   Expert Systems: Expert systems are AI programs designed to mimic the decision-making abilities of human experts in specific domains. They use knowledge, rules, and reasoning mechanisms to provide advice, solve problems, or make recommendations within their specialized areas.

2. How is a knowledgebase used in an expert system?
   Expert Systems: Expert systems are AI programs designed to mimic the decision-making abilities of human experts in specific domains. They use knowledge, rules, and reasoning mechanisms to provide advice, solve problems, or make recommendations within their specialized areas.

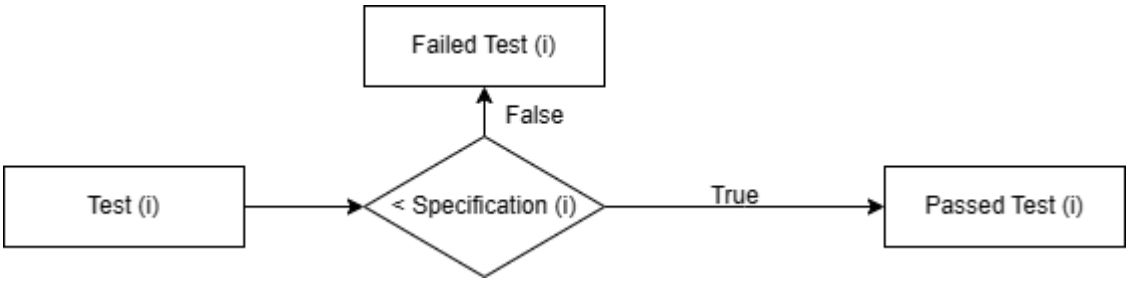
3. What are limitations of Expert systems?
   - Narrow Focus: Expert systems are confined to the domain they are designed for and lack generalization to other areas.
   - Knowledge Acquisition: Gathering accurate and comprehensive expert knowledge can be time-consuming and challenging.
   - Maintenance: Regular updates are necessary to keep the knowledge base up-to-date, which can be resource-intensive.
   - Lack of Common Sense: Expert systems may struggle with understanding context and common-sense reasoning.
   - Scalability: Adapting and scaling expert systems to handle a wide range of complex scenarios can be difficult.
   - Ethical Concerns: The decisions made by expert systems can have ethical implications, and ensuring fairness and accountability is a challenge.
   - Limited Learning: Expert systems often lack the ability to learn from new data or adapt to evolving situations.
   - High Development Costs: Building, maintaining, and fine-tuning expert systems can be expensive and time-consuming.

- Human Dependency: Overreliance on expert systems might reduce human decision-making skills and critical thinking.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Flowchart



Code:

Experiment 9

Problem Statement:

A manufacturing company is trying to improve their quality control process. They have a dataset of historical quality data, which includes the results of quality tests, as well as the specifications for each product. Implement a Simple Expert system to identify potential quality problems.

Product ID	Test 1 Result	Test 2 Result	Test 3 Result	Specification 1	Specification 2	Specification 3
1	95	20	8	100	25	10
2	98	22	9	100	20	10
3	93	18	7	95	15	8
4	100	24	10	100	25	12
5	96	21	8	98	20	10
6	92	19	6	95	15	8
7	90	17	5	95	15	8

Manual Rules:

Rule for Test 1:

- If the 'Test 1 Result' is less than 'Specification 1', flag as 'Test 1 Issue'.

Rule for Test 2:

- If the 'Test 2 Result' is less than 'Specification 2', flag as 'Test 2 Issue'.

Rule for Test 3:

- If the 'Test 3 Result' is less than 'Specification 3', flag as 'Test 3 Issue'.

Code:

```python
# importing required libraries
import numpy as np
import pandas as pd
```

[26] Python

Load the dataset into a DataFrame

```python
# Load the dataset into a DataFrame
def load_dataset():
    # TODO: Load the dataset from a CSV file or define it here as a dictionary and create a DataFrame.
    data = {
        'Product ID': [1, 2, 3, 4, 5, 6, 7],
        'Test 1 Result': [95, 98, 93, 100, 96, 92, 90],
        'Test 2 Result': [20, 22, 18, 24, 21, 19, 17],
        'Test 3 Result': [8, 9, 7, 10, 8, 6, 5],
        'Specification 1': [100, 100, 95, 100, 98, 95, 95],
        'Specification 2': [25, 20, 15, 25, 20, 15, 15],
        'Specification 3': [10, 10, 8, 12, 10, 8, 8]
    }

```

```
    df = pd.DataFrame(data)
    return df

```

[27] Python

Identify Quality problems for a single product

```
# Function to identify quality problems for a single product
def identify_quality_problems_for_product(product_id, test_results, specifications):
    problems = []
    for i in range(3):
        if test_results[i] < specifications[i]:
            problems.append(f"Product {product_id} failed Test {i+1}. Expected: {specifications[i]}, Got: {test_results[i]}")
    return problems
```

[28] Python

Identify Quality problems for entire dataset

```
# Function to identify quality problems in the entire dataset
def identify_quality_problems(df):
    problems = []
    for index, row in df.iterrows():
        product_id = row['Product ID']
        test_results = [row['Test 1 Result'], row['Test 2 Result'], row['Test 3 Result']]
        specifications = [row['Specification 1'], row['Specification 2'], row['Specification 3']]

        product_problems = identify_quality_problems_for_product(product_id, test_results, specifications)
        problems.extend(product_problems)
    return problems
```

[29] Python

Main function to solve the problem

```
# Main function to run the expert system
def main():
    df = load_dataset()
    quality_problems = identify_quality_problems(df)

    if len(quality_problems) == 0:
        print("No quality problems detected.")
    else:
        print("Quality problems detected:")
        for problem in quality_problems:
            print(problem)

if __name__ == "__main__":
    main()
```

[30] Python

```
Quality problems detected:
Product 1 failed Test 1. Expected: 100, Got: 95
Product 1 failed Test 2. Expected: 25, Got: 20
Product 1 failed Test 3. Expected: 10, Got: 8
Product 2 failed Test 1. Expected: 100, Got: 98
Product 2 failed Test 3. Expected: 10, Got: 9
Product 3 failed Test 1. Expected: 95, Got: 93
Product 3 failed Test 3. Expected: 8, Got: 7
Product 4 failed Test 2. Expected: 25, Got: 24
Product 4 failed Test 3. Expected: 12, Got: 10
Product 5 failed Test 1. Expected: 98, Got: 96
Product 5 failed Test 3. Expected: 10, Got: 8
Product 6 failed Test 1. Expected: 95, Got: 92
Product 6 failed Test 3. Expected: 8, Got: 6
Product 7 failed Test 1. Expected: 95, Got: 90
Product 7 failed Test 3. Expected: 8, Got: 5
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 10

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):

- Understand the widely known MYCIN and DART expert based systems.
- Implement an expert-based systems inspired by MYCIN and DART.

Outcome:

Students will be familiarized with basics of Fuzzy sets.

Problem Statement:

Create Python program for an expert-based systems inspired by MYCIN and DART for plant identification. The systems will take input from the user regarding observable characteristics of a plant and use a set of rules and certainty factors to identify the most likely plant species.
Use the following dataset for setting up the rules:

Plant ID	Leaf Shape	Flower Color	Species
1	Lobed	Pink	Rose
2	Lobed	Pink	Cherry Blossom
3	Oval	Yellow	Daffodil
4	Oval	Yellow	Sunflower
5	Palmate	White	Magnolia
6	Palmate	White	Dogwood
7	Oval	Red	Tulip
8	Oval	Red	Poppy

Background Study:

MYCIN, a medical diagnosis system, utilized rule-based reasoning with certainty factors to diagnose infectious diseases and suggest treatments based on patient symptoms. Meanwhile, DART excelled in diagnosing

complex electronic circuit faults using model-based reasoning and causal analysis. These pioneering systems demonstrated the effectiveness of knowledge-based approaches in real-world problem-solving, setting the foundation for the development of the expert-based system aimed at identifying potential quality problems in the manufacturing company's quality control process for plant identification.

Question Bank:

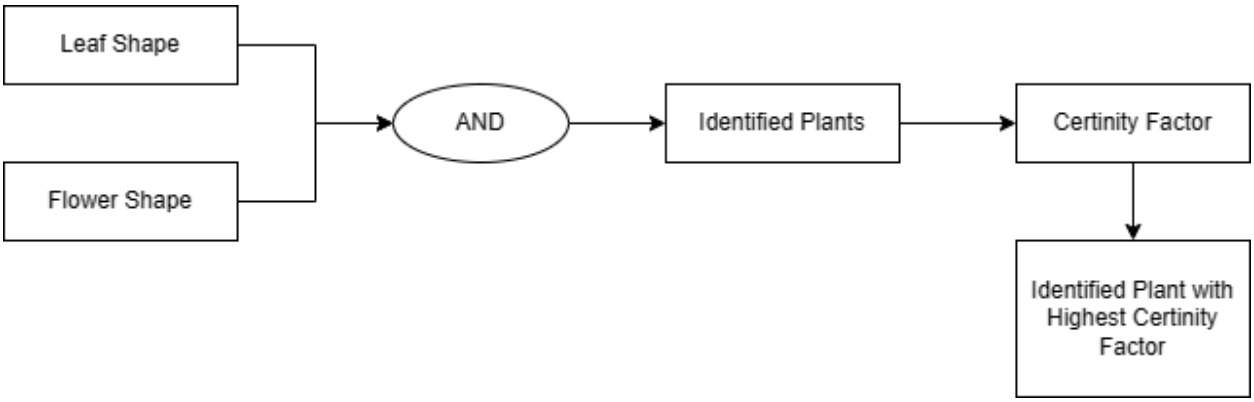
1. How does MYCIN and DART based Expert system work?
   - **MYCIN:** MYCIN was an early expert system used for medical diagnosis of infectious diseases. It worked by collecting patient symptoms and medical data, applying a set of rules based on expert knowledge in the medical field, and then providing a diagnosis and treatment recommendation.
   - **DART:** DART (Diagnostic and Reasoning Tool) is another medical expert system that focuses on diagnosing diseases based on clinical symptoms and laboratory results. It uses an inference engine to match symptoms with known patterns and suggest possible diagnoses.

2. What are some of the challenges faced while building such systems?
   - Knowledge Acquisition: Extracting expert knowledge and translating it into a machine-readable format can be difficult and time-consuming.
   - Knowledge Representation: Choosing the right representation for knowledge that captures both declarative and procedural aspects.
   - Rule Management: Handling large rule sets and ensuring consistency, accuracy, and maintainability.
   - Inference Mechanisms: Designing efficient and accurate inference engines for drawing conclusions from knowledge.
   - Domain Limitations: Expert systems are often limited to specific domains and may struggle with handling unfamiliar situations.
   - Common Sense Reasoning: Infusing systems with human-like common-sense reasoning remains a challenge.
   - User Interaction: Designing intuitive and effective user interfaces and explanation mechanisms.
   - Continuous Learning: Ensuring expert systems can adapt and learn from new data and experiences.
   - Ethical Considerations: Addressing ethical concerns, accountability, and biases in decision-making.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Flowchart



Code:

Experiment 10

Problem Statement:

Create Python program for an expert-based systems inspired by MYCIN and DART for plant identification. The systems will take input from the user regarding observable characteristics of a plant and use a set of rules and certainty factors to identify the most likely plant species. Use the following dataset for setting up the rules:

Plant ID	Leaf Shape	Flower Color	Species	Certainty Factor (shape,color)
1	Lobed	Pink	Rose	(0.7, 0.8)
2	Lobed	Pink	Cherry Blossom	(0.7, 0.7)
3	Oval	Yellow	Daffodil	(0.8, 0.9)
4	Oval	Yellow	Sunflower	(0.8, 0.85)
5	Palmate	White	Magnolia	(0.75, 0.7)
6	Palmate	White	Dogwood	(0.75, 0.75)
7	Oval	Red	Tulip	(0.8, 0.8)
8	Oval	Red	Poppy	(0.8, 0.75)

Code:

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

[22]
```python
# creating  dataframe from given data
def create_dataframe():
    data = {
        'Plant ID': [1, 2, 3, 4, 5, 6, 7, 8],
        'Leaf Shape': ['Lobed', 'Lobed', 'Oval', 'Oval', 'Palmate', 'Palmate', 'Oval', 'Oval'],
        'Flower Color': ['Pink', 'Pink', 'Yellow', 'Yellow', 'White', 'White', 'Red', 'Red'],
        'Species': ['Rose', 'Cherry Blossom', 'Daffodil', 'Sunflower', 'Magnolia', 'Dogwood', 'Tulip', 'Poppy'],
        'Certainty Factor (shape,color)': [(0.7, 0.8), (0.7, 0.7), (0.8, 0.9), (0.8, 0.85), (0.75, 0.7), (0.75, 0.75), (0.8, 0.8), (0.8, 0.75)]
    }
    return pd.DataFrame(data)
```
Python

[23]
```python
# function to calculate certainty factor
def calculate_certainty(user_shape, user_color, plant):
    shape_cf, color_cf = plant['Certainty Factor (shape,color)']
    return (shape_cf if plant['Leaf Shape'] == user_shape else 0) * (color_cf if plant['Flower Color'] == user_color else 0)

# function to identify plant
def identify_plant(df, user_shape, user_color):
    df['Calculated Certainty'] = df.apply(
        lambda plant: calculate_certainty(user_shape, user_color, plant), axis=1)
    max_certainty = df['Calculated Certainty'].max()
    identified_plants = df[df['Calculated Certainty'] == max_certainty]
    return identified_plants, max_certainty
```
Python

```python
# function to identify plant
def plant_identification(df, leaf_shape, flower_color):
    identified_plants, max_certainty = identify_plant(
        df, leaf_shape, flower_color)
    if max_certainty > 0:
        for _, plant in identified_plants.iterrows():
            print(
                f"The identified plant is {plant['Species']}. ")
    else:
        print("No matching plant found.")
```

[24] Python

```python
# main function
if __name__ == "__main__":
    df = create_dataframe()
    print("Plant Identification System")
    leaf_shape = input(
        "Enter leaf shape (Lobed, Oval, Palmate): ").capitalize()
    flower_color = input(
        "Enter flower color (Pink, Yellow, White, Red): ").capitalize()
    plant_identification(df, leaf_shape, flower_color)
```

[25] Python

```
Plant Identification System
Enter leaf shape (Lobed, Oval, Palmate): Palmate
Enter flower color (Pink, Yellow, White, Red): White
The identified plant is Dogwood.
```

THE NORTHCAP UNIVERSITY

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.

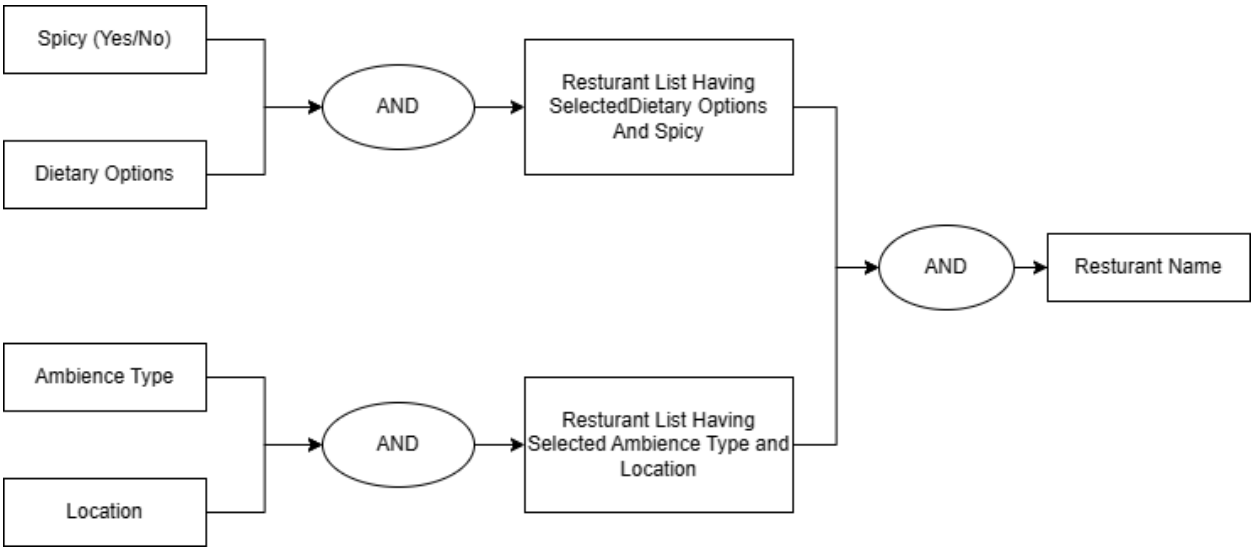
   Limitations:

- **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']

```

```python
    # 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.
```

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 12

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: 18.11.2023
Faculty Signature:
Grade:

Objective(s):

- Understand Feature Map, Data Cleaning and Visualization steps.
- Study about the importance of Data Visualization step before implementing ML models.

Outcome:

- Student will be familiarized with the important steps of Data Cleaning and Visualization.

Problem Statement:

Given a simple dataset containing information about a group of students, including the number of hours they studied and their scores in a previous exam. Perform data cleaning, selection, and visualization to gain insights into the relationships between study hours, exam scores, and student pass/fail outcomes.

Student ID	Study Hours	Exam Score	Pass/Fail
1	5	85	Pass
2	3	123	Fail
3	6	90	Pass
4	2	40	Fail
5	4	78	Pass
6	7	95	Pass
7	1	30	Fail
	5	80	Pass

Background Study:

Data cleaning is the crucial process of identifying and rectifying errors, inconsistencies, and missing values in a dataset to ensure accuracy and reliability. Visualization, through graphs and

charts, plays a pivotal role in presenting data in a concise and easily understandable format. Together, data cleaning and visualization enhance data quality, facilitate pattern recognition, and empower informed decision-making, making them essential steps in any data analysis or modelling task.

Question Bank:

1. What are steps involved in Data Cleaning?

   Data cleaning involves preparing and organizing raw data for analysis. The steps typically include:

   - Handling Missing Values: Identifying and dealing with missing data, either by imputing values or removing incomplete records.
   - Handling Outliers: Detecting and addressing outliers that can skew analysis or model performance.
   - Dealing with Duplicates: Identifying and removing duplicated records to prevent duplication bias.
   - Normalization and Scaling: Scaling numerical features to comparable ranges to avoid dominance of certain features.
   - Encoding Categorical Variables: Converting categorical data into numerical format suitable for analysis.
   - Handling Noisy Data: Reducing noise caused by errors or inconsistencies in the data.
   - Addressing Data Integrity: Ensuring data integrity by validating data against predefined rules.
   - Data Transformation: Performing transformations like logarithm or square root to make data more suitable for analysis.
   - Feature Selection: Selecting relevant features for analysis to avoid dimensionality issues.
   - Data Formatting: Ensuring consistent data formats and units across the dataset.

2. How does Machine Learning Algorithms work?

   Machine learning algorithms work by learning patterns and relationships from data in order to make predictions or decisions. The general process involves:

   - Data Collection: Gathering relevant and representative data for the problem.
   - Data Preprocessing: Cleaning, transforming, and preparing data for analysis.
   - Feature Extraction/Selection: Choosing informative features that represent the data well.
   - Model Selection: Choosing an appropriate machine learning algorithm based on the problem (classification, regression, clustering, etc.).
   - Model Training: Feeding the algorithm with labeled data to learn patterns and adjust internal parameters.
   - Model Evaluation: Assessing the model's performance using metrics like accuracy, precision, recall, etc.

- Hyperparameter Tuning: Adjusting hyperparameters to optimize the model's performance.
- Model Deployment: Using the trained model to make predictions on new, unseen data.

3. What is a Feature map and how can it be used?

   A feature map, in the context of deep learning and neural networks, refers to the output of a specific layer in a network after applying a set of learned filters or convolutional kernels to the input data. Each feature map captures specific features or patterns from the input. Feature maps from different layers capture different levels of abstraction.

   In convolutional neural networks (CNNs), feature maps play a crucial role in learning hierarchical representations of input data. These learned features can be used for tasks like image classification, object detection, and segmentation. The feature maps closer to the input layer capture low-level features like edges and textures, while those deeper in the network capture higher-level features like object parts and complex patterns. By using these learned features, neural networks can effectively understand and analyze complex data like images and text.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 12

Problem Statement:

Perform Data Quality Checks and various Profiling Functions over the UCI Adult Dataset.

Dataset Link:

http://archive.ics.uci.edu/dataset/2/adult

Installing Dependencies:

In [ ]:
```
! pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in c:\users\mainp\appdata\local\packages\py
thonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311
\site-packages (0.0.3)
```

Code:

In [ ]:
```
# importing required libraries
import pandas as pd
import numpy as np
from ucimlrepo import fetch_ucirepo
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

In [ ]:
```
# loading the dataset into a pandas dataframe
def load_dataset():
    # TODO: Load the dataset from using ucimlrepo module as provided in the dataset
    # Define the dataset as a Pandas DataFrame
    # fetch dataset
    adult = fetch_ucirepo(id=2)

    # data (as pandas dataframes)
    X = adult.data.features
    y = adult.data.targets

    df = pd.concat([X, y], axis=1)

    return df
```

In [ ]:
```
def data_quality_checks(df):
    # handling missing values
    print("\nNumber of missing values in each column: ")
    print(df.isnull().sum())
    # df = df.dropna() # removing missing values
```

```
    print("\n")

    # check for duplicates
    print("\nNumber of duplicates in the dataset: ")
    print(df.duplicated().sum())
    # df = df.drop_duplicates() # removing duplicates

    print("\n")

    # check for unique values in categorical columns
    print("\nUnique values in categorical columns: ")
    print(df.select_dtypes(include=['object']).nunique())

    print("\n")

    # check for outliers using z-score
    print("\nNumber of outliers in each column: ")
    z = np.abs(stats.zscore(df.select_dtypes(include=['int64', 'float64'])))
    print(np.where(z > 3)[0].size)
    # df = df[(z < 3).all(axis=1)] # removing outliers

    # returning the updated dataframe
    return df
```

In [ ]:

```
def profiling_visualization(df):
    # display basic information about the dataset
    print("\nBasic information about the dataset:")
    print(df.info())

    print("\n")

    # descriptive statistics
    print("\nDescriptive statistics:")
    print(df.describe())

    # categorical feature profiling
    print("\nCategorical feature profiling:")
    print("Data type of each column: ")
    print(df.select_dtypes(include=['object']).describe())
    print("\nUnique values in each column: ")
    print(df.select_dtypes(include=['object']).nunique())
    print("\nDistribution of values in each column: ")
    for col in df.select_dtypes(include=['object']).columns:
        print("\n", col, ":")
        value_counts = df[col].value_counts()
        value_counts_other = value_counts[value_counts >= len(df) * 0.01]
        value_counts_other['Other'] = value_counts[value_counts < len(
            df) * 0.01].sum()
        print(value_counts_other)
    # plotting pie charts for categorical features
    print("\nPie Charts for categorical features:")
    for col in df.select_dtypes(include=['object']).columns:
        plt.figure(figsize=(5, 5))
        plt.title(col)
        value_counts = df[col].value_counts()
        value_counts_other = value_counts[value_counts >= len(df) * 0.01]
```

```
        value_counts_other['Other'] = value_counts[value_counts < len(
            df) * 0.01].sum()
        value_counts_other.plot.pie(autopct='%1.1f%%')
        plt.show()

    # numerical feature profiling
    print("\nNumerical feature profiling:")
    print("Statistical summary of each column: ")
    print(df.select_dtypes(include=['int64', 'float64']).describe())
    print("\nDistribution of values in numerical column: ")
    for col in df.select_dtypes(include=['int64', 'float64']).columns:
        print("\n", col, ":")
        print(df[col].value_counts())
    # plotting histograms for numerical features
    print("\nHistograms for numerical features:")
    df.hist(figsize=(10, 10), grid=False, color='green', bins=20)
    plt.show()
    # plotting correlation matrix for numerical features
    print("\nCorrelation matrix for numerical features:")
    plt.figure(figsize=(10, 10))
    sns.heatmap(df.select_dtypes(
        include=['int64', 'float64']).corr(), annot=True, cmap='Blues')
```

In [ ]:
```
# load the dataset
df = load_dataset()
print("\nFirst 5 rows of the dataset: ")
print(df.head().to_markdown)
```

```
First 5 rows of the dataset:
<bound method DataFrame.to_markdown of    age         workclass  fnlwgt  education
education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13

       marital-status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2           Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week native-country income
0          2174             0              40  United-States  <=50K
1             0             0              13  United-States  <=50K
2             0             0              40  United-States  <=50K
3             0             0              40  United-States  <=50K
4             0             0              40           Cuba  <=50K  >
```

In [ ]:
```
print("\n\n")

# data quality checks
print("\nData quality checks: ")
data_quality_checks(df)
```

```
print("\n\n")

# profiling and visualization
print("\nProfiling and visualization: ")
profiling_visualization(df)
```

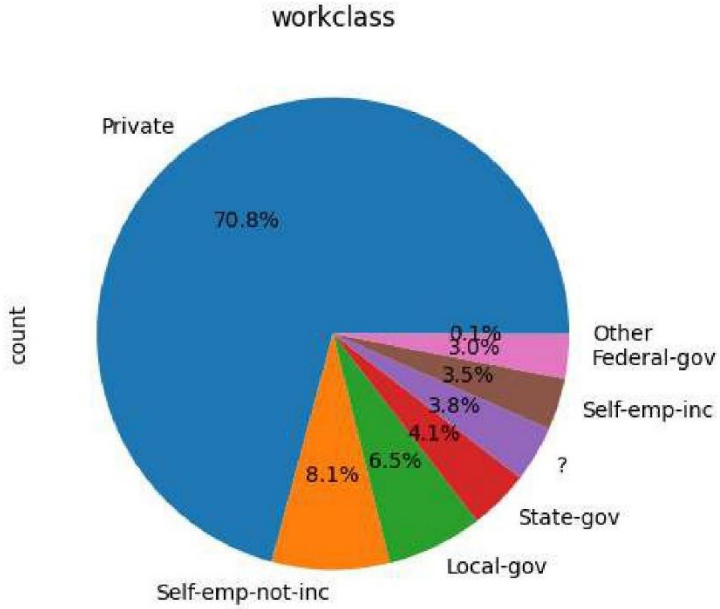
```
Data quality checks:

Number of missing values in each column:
age                0
workclass        963
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation       966
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   274
income             0
dtype: int64


Number of duplicates in the dataset:
29


Unique values in categorical columns:
workclass          9
education         16
marital-status     7
occupation        15
relationship       6
race               5
sex                2
native-country    42
income             4
dtype: int64


Number of outliers in each column:
4250



Profiling and visualization:

Basic information about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
```

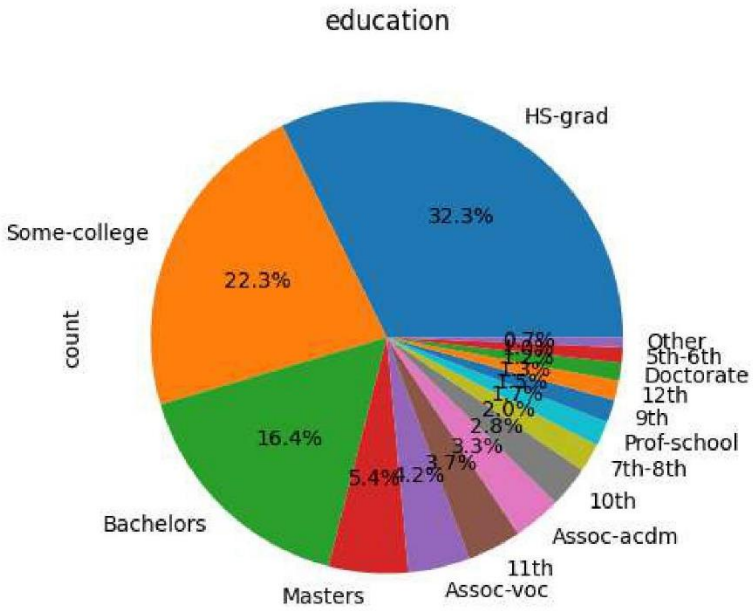
```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       47879 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64
 5   marital-status  48842 non-null  object
 6   occupation      47876 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64
 11  capital-loss    48842 non-null  int64
 12  hours-per-week  48842 non-null  int64
 13  native-country  48568 non-null  object
 14  income          48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
None



Descriptive statistics:
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  48842.000000  4.884200e+04   48842.000000  48842.000000  48842.000000
mean      38.643585  1.896641e+05      10.078089   1079.067626     87.502314
std       13.710510  1.056040e+05       2.570973   7452.019058    403.004552
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.175505e+05       9.000000      0.000000      0.000000
50%       37.000000  1.781445e+05      10.000000      0.000000      0.000000
75%       48.000000  2.376420e+05      12.000000      0.000000      0.000000
max       90.000000  1.490400e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    48842.000000
mean        40.422382
std         12.391444
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000

Categorical feature profiling:
Data type of each column:
       workclass education      marital-status      occupation relationship  \
count      47879     48842               48842           47876        48842
unique         9        16                   7              15            6
top      Private   HS-grad  Married-civ-spouse  Prof-specialty      Husband
freq       33906     15784               22379            6172        19716

         race    sex native-country income
count   48842  48842          48568  48842
unique      5      2             42      4
top     White   Male  United-States  <=50K
```

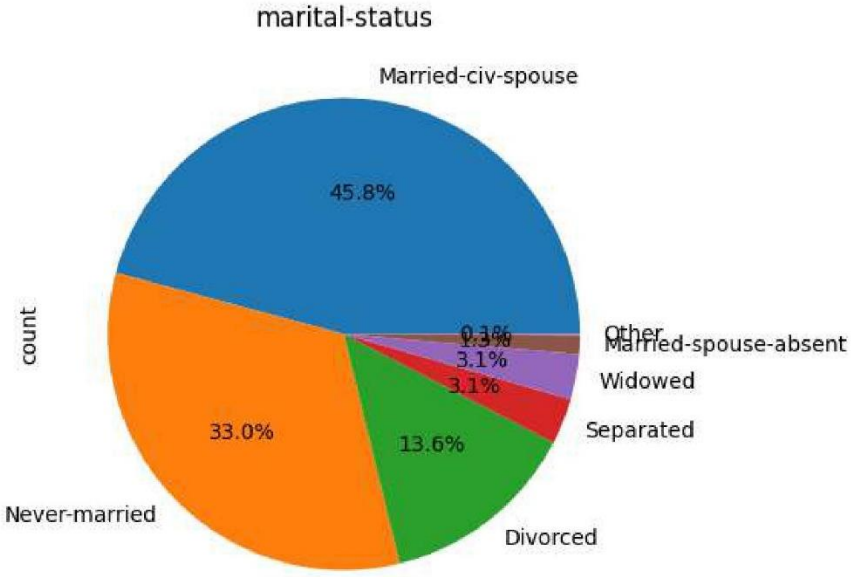
```
freq    41762  32650          43832  24720

Unique values in each column:
workclass          9
education         16
marital-status     7
occupation        15
relationship       6
race               5
sex                2
native-country    42
income             4
dtype: int64

Distribution of values in each column:

 workclass :
workclass
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
State-gov            1981
?                    1836
Self-emp-inc         1695
Federal-gov          1432
Other                  31
Name: count, dtype: int64

 education :
education
HS-grad         15784
Some-college    10878
Bachelors        8025
Masters          2657
Assoc-voc        2061
11th             1812
Assoc-acdm       1601
10th             1389
7th-8th           955
Prof-school       834
9th               756
12th              657
Doctorate         594
5th-6th           509
Other             330
Name: count, dtype: int64

 marital-status :
marital-status
Married-civ-spouse       22379
Never-married            16117
Divorced                  6633
Separated                 1530
Widowed                   1518
Married-spouse-absent      628
Other                       37
```

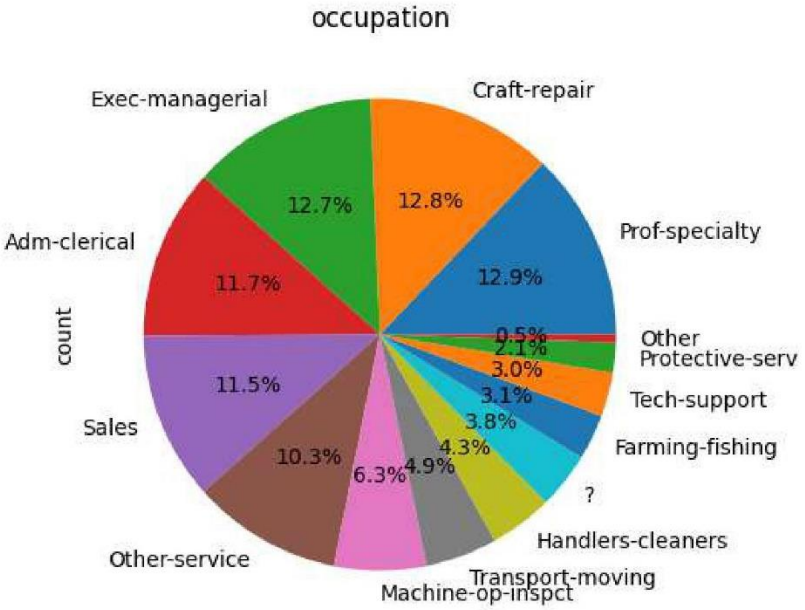
```
Name: count, dtype: int64

 occupation :
occupation
Prof-specialty       6172
Craft-repair         6112
Exec-managerial      6086
Adm-clerical         5611
Sales                5504
Other-service        4923
Machine-op-inspct    3022
Transport-moving     2355
Handlers-cleaners    2072
?                    1843
Farming-fishing      1490
Tech-support         1446
Protective-serv       983
Other                 257
Name: count, dtype: int64

 relationship :
relationship
Husband           19716
Not-in-family     12583
Own-child          7581
Unmarried          5125
Wife               2331
Other-relative     1506
Other                 0
Name: count, dtype: int64

 race :
race
White                41762
Black                 4685
Asian-Pac-Islander    1519
Other                  876
Name: count, dtype: int64

 sex :
sex
Male      32650
Female    16192
Other         0
Name: count, dtype: int64

 native-country :
native-country
United-States    43832
Mexico             951
?                  583
Other             3202
Name: count, dtype: int64

 income :
income
```

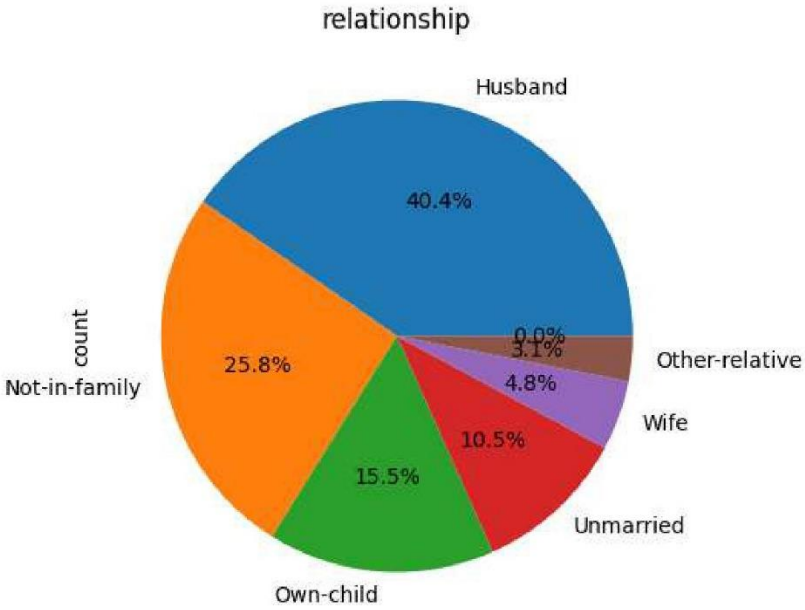
```
<=50K     24720
<=50K.    12435
>50K       7841
>50K.      3846
Other         0
Name: count, dtype: int64

Pie Charts for categorical features:
```

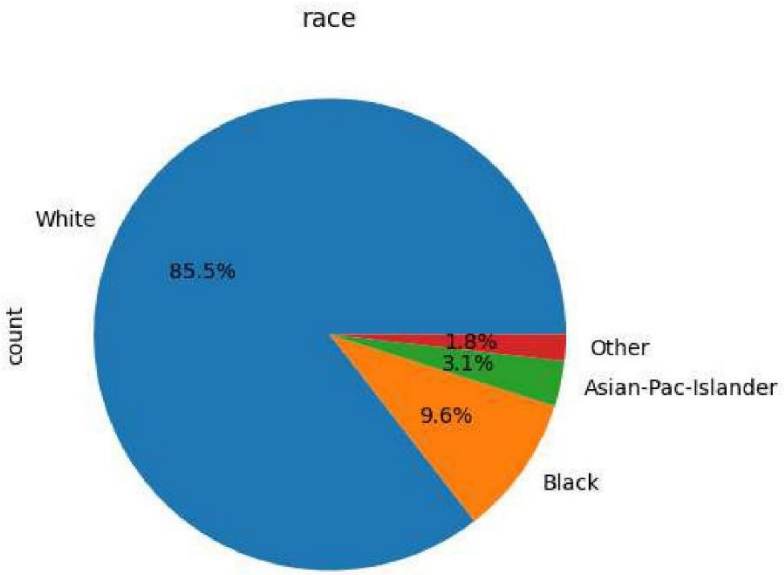


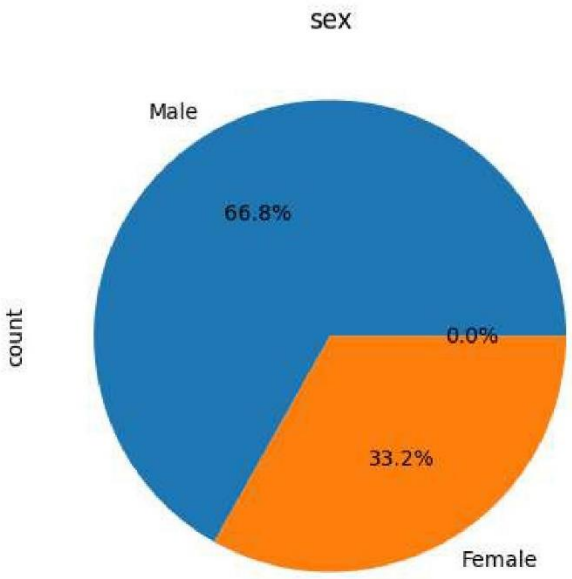

education
HS-grad
32.3%
Some-college
22.3%
count
16.4%
Bachelors
5.4%
Masters
4.2%
Assoc-voc
3.7%
11th
3.3%
Assoc-acdm
2.8%
10th
2.0%
7th-8th
1.7%
Prof-school
1.5%
9th
1.3%
12th
Doctorate
5th-6th
Other

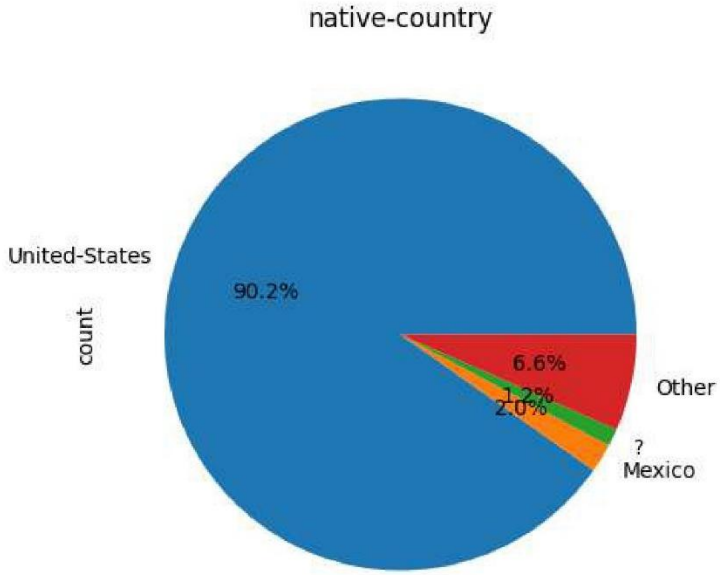

marital-status
Married-civ-spouse
45.8%
count
33.0%
Never-married
13.6%
Divorced
3.1%
Separated
3.1%
Widowed
Married-spouse-absent
Other

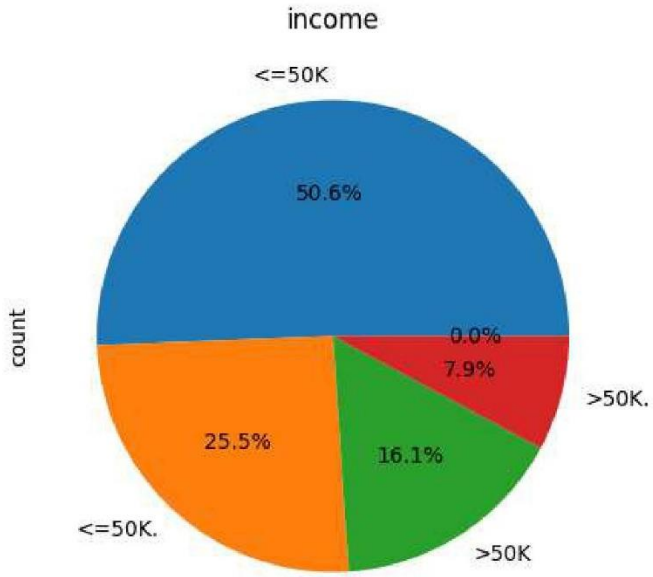

occupation
count
Exec-managerial
12.7%
Craft-repair
12.8%
Prof-specialty
12.9%
Adm-clerical
11.7%
Other
Protective-serv
Tech-support
Farming-fishing
?
Handlers-cleaners
Transport-moving
Machine-op-inspct
Other-service
Sales
11.5%
10.3%
6.3%
4.9%
4.3%
3.8%
3.1%
3.0%
2.1%
0.5%


relationship
count
Husband
40.4%
Not-in-family
25.8%
Other-relative
3.1%
Wife
4.8%
Unmarried
10.5%
Own-child
15.5%
0.0%


race
White
85.5%
count
1.8%
Other
3.1%
Asian-Pac-Islander
9.6%
Black


sex
Male
66.8%
count
0.0%
33.2%
Female

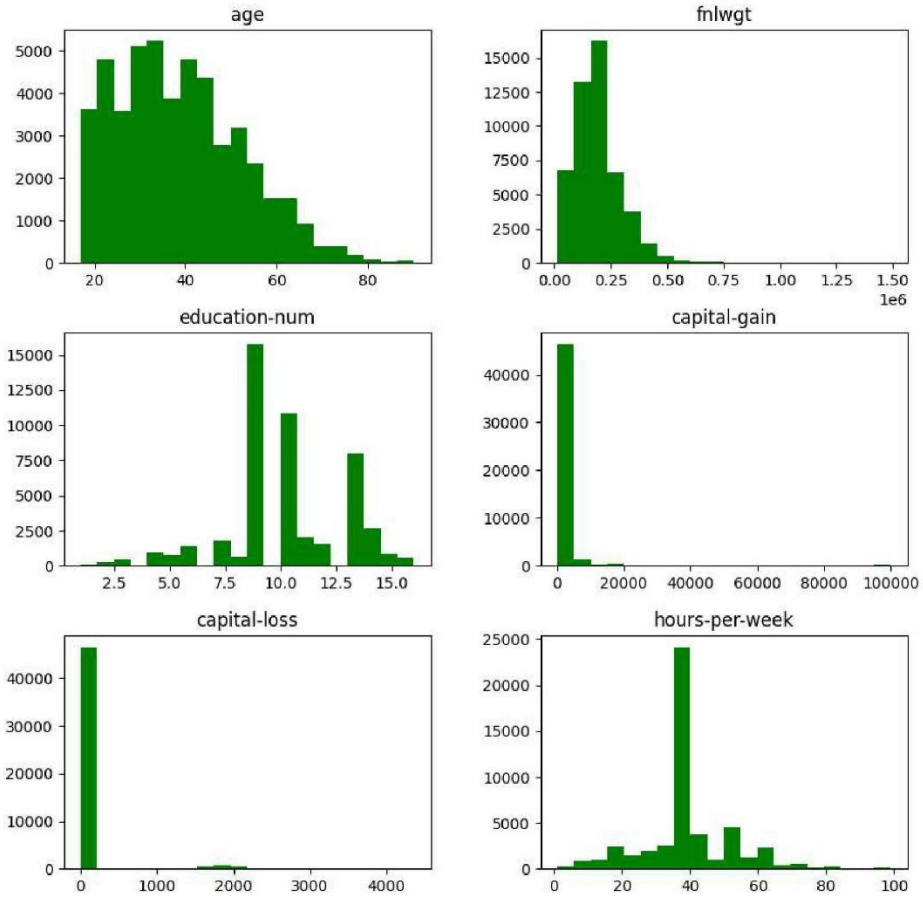

native-country
United-States
90.2%
count
6.6%
Other
1.2%
2.0%
?
Mexico


income
<=50K
50.6%
count
0.0%
7.9%
>50K.
25.5%
16.1%
<=50K.
>50K

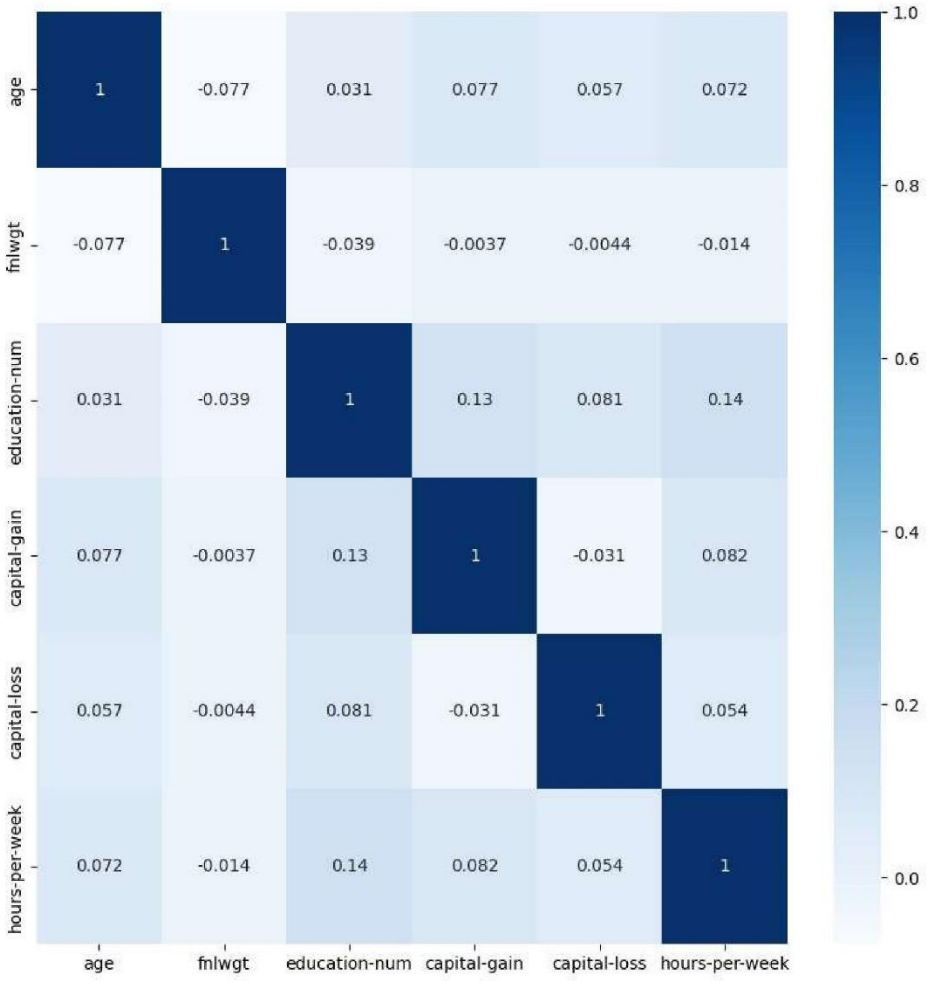
```
Numerical feature profiling:
Statistical summary of each column:
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  48842.000000  4.884200e+04   48842.000000  48842.000000  48842.000000
mean      38.643585  1.896641e+05      10.078089   1079.067626     87.502314
std       13.710510  1.056040e+05       2.570973   7452.019058    403.004552
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.175505e+05       9.000000      0.000000      0.000000
50%       37.000000  1.781445e+05      10.000000      0.000000      0.000000
75%       48.000000  2.376420e+05      12.000000      0.000000      0.000000
max       90.000000  1.490400e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    48842.000000
mean        40.422382
std         12.391444
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000

Distribution of values in numerical column:

 age :
age
36    1348
35    1337
33    1335
23    1329
31    1325
      ...
88       6
85       5
87       3
89       2
86       1
Name: count, Length: 74, dtype: int64

 fnlwgt :
fnlwgt
203488    21
120277    19
190290    19
125892    18
126569    18
          ..
286983     1
185942     1
234220     1
214706     1
350977     1
Name: count, Length: 28523, dtype: int64

 education-num :
education-num
```

```
9     15784
10    10878
13     8025
14     2657
11     2061
7      1812
12     1601
6      1389
4       955
15      834
5       756
8       657
16      594
3       509
2       247
1        83
Name: count, dtype: int64

 capital-gain :
capital-gain
0        44807
15024      513
7688       410
7298       364
99999      244
          ...
22040        1
2387         1
1639         1
1111         1
6612         1
Name: count, Length: 123, dtype: int64

 capital-loss :
capital-loss
0       46560
1902      304
1977      253
1887      233
2415       72
         ...
1539        1
1870        1
1911        1
2465        1
1421        1
Name: count, Length: 99, dtype: int64

 hours-per-week :
hours-per-week
40    22803
50     4246
45     2717
60     2177
35     1937
      ...
```

```
79        1
94        1
82        1
87        1
69        1
Name: count, Length: 96, dtype: int64

Histograms for numerical features:
```




```
Correlation matrix for numerical features:
```


age
fnlwgt
education-num
capital-gain
capital-loss
hours-per-week
1
-0.077
0.031
0.077
0.057
0.072
-0.077
1
-0.039
-0.0037
-0.0044
-0.014
0.031
-0.039
1
0.13
0.081
0.14
0.077
-0.0037
0.13
1
-0.031
0.082
0.057
-0.0044
0.081
-0.031
1
0.054
0.072
-0.014
0.14
0.082
0.054
1
1.0
0.8
0.6
0.4
0.2
0.0

EXPERIMENT NO. 13

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: 25.11.2023
Faculty Signature:
Grade:

Objective(s):

- Understand and study Support Vector Machines for classification problems.
- Study about how Anomaly Detection can be performed using supervised learning.

Outcome:
Students will be familiarized with Support Vector Machine for classification.

Problem Statement:

Python program to implement Credit Card Fraud detection using Support Vector Machine classification.

URL for the dataset:

https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud

Background Study:

SVMs are machine learning algorithms used for classification and regression. In anomaly detection, SVMs serve as one-class classifiers, identifying anomalies as data points deviating from the learned normal representation. They create a decision boundary to separate normal data from anomalies, making them effective in detecting novel or rare instances outside the normal data distribution. SVM-based anomaly detection is useful when labeled anomaly data is scarce, making it applicable in domains like fraud and intrusion detection.

Question Bank:

1. What is a Support Vector Machine?
   A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane that best separates

different classes or predicts a target value while maximizing the margin (distance) between the hyperplane and the closest data points (support vectors).

2. How SVM is used to solve problems?
   SVM is used to solve problems by identifying the optimal hyperplane that separates different classes or predicts values for regression tasks. It involves selecting the hyperplane that maximizes the margin between support vectors while minimizing classification errors. SVM can also employ kernel functions to transform data into higher-dimensional spaces, allowing it to handle non-linear separations.

3. List out the advantages and disadvantages of SVM on Anomaly detection?
   Advantages of SVM for Anomaly Detection
   - Effective in High-Dimensional Spaces: SVM works well in high-dimensional feature spaces, which is beneficial for capturing complex relationships in anomaly data.
   - Ability to Handle Unbalanced Data: SVM can be adjusted to handle unbalanced datasets, which is common in anomaly detection scenarios.
   - Robust to Outliers: SVM's focus on support vectors makes it less sensitive to outliers, which is important in anomaly detection where outliers represent anomalies.

   Disadvantages of SVM for Anomaly Detection:
   - Sensitivity to Hyperparameters: SVM's performance is influenced by hyperparameters like the choice of kernel and regularization parameters, which may require careful tuning.
   - Computationally Intensive: Training SVMs can be computationally intensive, especially for large datasets or when using non-linear kernels.
   - Difficulty with Large Datasets: SVM's efficiency diminishes with larger datasets due to the need to store and compute kernel matrices.
   - Limited Interpretability: SVMs provide decision boundaries without inherent feature importance scores, making it less interpretable compared to some other algorithms.
   - Keep in mind that the effectiveness of SVM for anomaly detection depends on the specific characteristics of the data and the problem at hand.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 13

Problem Statement:

Python program to implement Credit Card Fraud detection using Support Vector Machine classification.

URL for the dataset: https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud

Install Dependencies:

In [ ]:
```
! pip install tabulate
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python
311\lib\site-packages (0.9.0)
```

Code:

In [ ]:
```
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as snss

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score, precisi
```

In [ ]:
```
# importing the dataset
dataset = pd.read_csv('card_transdata_dataset.csv')
print(dataset.head().to_markdown())
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_price	repeat_retailer	used_chip	used_pin_number	online_order	fraud
0	57.8779	0.31114	1.94594	1	1	0	0	0
1	10.8299	0.175592	1.29422	1	0	0	0	0
2	5.09108	0.805153	0.427715	1	0	0	1	0
3	2.24756	5.60004	0.362663	1	1	0	1	0
4	44.1909	0.566486	2.22277	1	1	0	1	0

In [ ]:
```
```

In [ ]:
```
# checking for null values
print("\nChecking for null values:")
print(dataset.isnull().sum())

# dropping null values

dataset = dataset.dropna()
print(dataset.isnull().sum)
```

```
Checking for null values:
distance_from_home               0
distance_from_last_transaction   0
ratio_to_median_purchase_price   0
repeat_retailer                  0
used_chip                        0
used_pin_number                  0
online_order                     0
fraud                            0
dtype: int64
<bound method NDFrame._add_numeric_operations.<locals>.sum of        distance_from_home  dist
ance_from_last_transaction  \
0                    False                           False
1                    False                           False
2                    False                           False
3                    False                           False
4                    False                           False
...                    ...                             ...
999995               False                           False
999996               False                           False
999997               False                           False
999998               False                           False
999999               False                           False

        ratio_to_median_purchase_price  repeat_retailer  used_chip  \
0                                False            False      False
1                                False            False      False
2                                False            False      False
3                                False            False      False
4                                False            False      False
...                                ...              ...        ...
999995                           False            False      False
999996                           False            False      False
999997                           False            False      False
999998                           False            False      False
999999                           False            False      False

        used_pin_number  online_order  fraud
0                 False         False  False
1                 False         False  False
2                 False         False  False
3                 False         False  False
4                 False         False  False
...                 ...           ...    ...
999995            False         False  False
999996            False         False  False
999997            False         False  False
999998            False         False  False
999999            False         False  False

[1000000 rows x 8 columns]>
```

In [ ]:
```
# checking for duplicate values
print("\nChecking for duplicate values:")
print(dataset.duplicated().sum())
```

```
Checking for duplicate values:
0
```

In [ ]:
```
# checking for outliers
print("\nChecking for outliers:")
print(dataset.describe())
```

```
Checking for outliers:
       distance_from_home  distance_from_last_transaction  \
count      1000000.000000                  1000000.000000
mean            26.628792                        5.036519
std             65.390784                       25.843093
min              0.004874                        0.000118
25%              3.878008                        0.296671
50%              9.967760                        0.998650
75%             25.743985                        3.355748
max          10632.723672                    11851.104565

       ratio_to_median_purchase_price  repeat_retailer       used_chip  \
count                 1000000.000000   1000000.000000  1000000.000000
mean                        1.824182         0.881536        0.350399
std                         2.799589         0.323157        0.477095
min                         0.004399         0.000000        0.000000
25%                         0.475673         1.000000        0.000000
50%                         0.997717         1.000000        0.000000
75%                         2.096370         1.000000        1.000000
max                       267.802942         1.000000        1.000000

       used_pin_number    online_order           fraud
count   1000000.000000  1000000.000000  1000000.000000
mean          0.100608        0.650552        0.087403
std           0.300809        0.476796        0.282425
min           0.000000        0.000000        0.000000
25%           0.000000        0.000000        0.000000
50%           0.000000        1.000000        0.000000
75%           0.000000        1.000000        0.000000
max           1.000000        1.000000        1.000000
```

In [ ]:
```
# checking the number of fraud and non-fraud transactions
print("\nNumber of fraud and non-fraud transactions:")
print(dataset['fraud'].value_counts())
```

```
Number of fraud and non-fraud transactions:
fraud
0.0    912597
1.0     87403
Name: count, dtype: int64
```

In [ ]:
```
# visualizing the number of fraud and non-fraud transactions
fraud_data = dataset[dataset['fraud'] == 1]
non_fraud_data = dataset[dataset['fraud'] == 0]

# calculating the minimum number of fraud or non-fraud transactions
min_instances = min(len(fraud_data), len(non_fraud_data))
```

In [ ]:
```
# creating a subset of the data with equal number of fraud and non-fraud transactions
fraud_subset = fraud_data.sample(n=20000, random_state=42)
non_fraud_subset = non_fraud_data.sample(n=20000, random_state=42)
```

In [ ]:
```
# concatenating the fraud and non-fraud subsets
small_dataset = pd.concat([fraud_subset, non_fraud_subset])
```

In [ ]:
```
# scaling the dataset
sc = StandardScaler()
sc.fit(small_dataset.drop('fraud', axis=1))
```

Out[ ]:
```
▾ StandardScaler

StandardScaler()
```

In [ ]:
```
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(small_dataset.drop(
    'fraud', axis=1), small_dataset['fraud'], test_size=0.2, random_state=42)
```

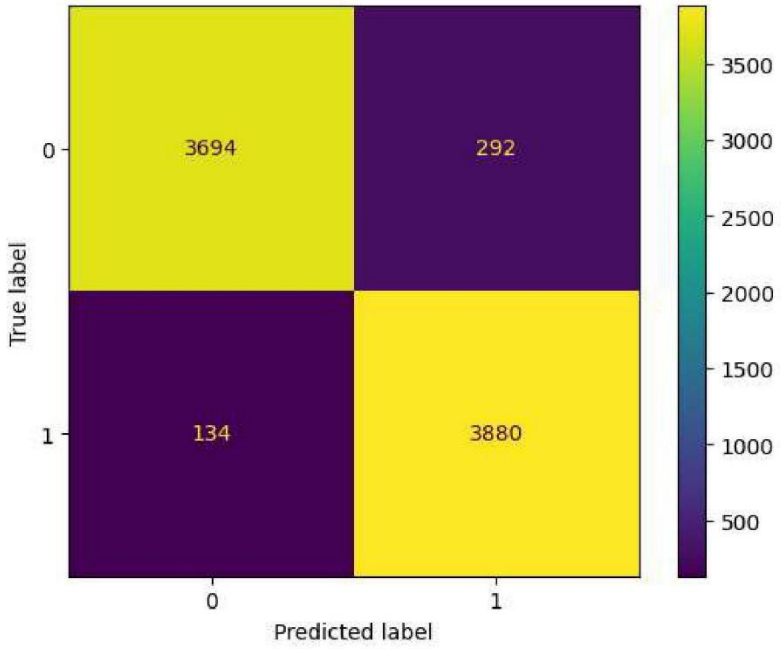
In [ ]:
```
# training the model using svm and changing the kernel to poly
classifier = SVC(kernel='linear', random_state=0)
# classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)
```

Out[ ]:
```
SVC(kernel='linear', random_state=0)
```

In [ ]:
```
# predicting the test set results
y_pred = classifier.predict(X_test)
```

In [ ]:
```
# calculating the metrics
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)

print("\nConfusion Matrix Display:")
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))

print("\nPrecision Score:")
print(precision_score(y_test, y_pred))

print("\nRecall Score:")
print(recall_score(y_test, y_pred))

print("\nF1 Score:")
print(f1_score(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[3694  292]
 [ 134 3880]]

Confusion Matrix Display:

Accuracy Score:
0.94675

Precision Score:
0.9300095877277086

Recall Score:
0.966616841056303

F1 Score:
0.9479599315905204

Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      0.93      0.95      3986
         1.0       0.93      0.97      0.95      4014

    accuracy                           0.95      8000
   macro avg       0.95      0.95      0.95      8000
weighted avg       0.95      0.95      0.95      8000
```



THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 15

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: 25.11.2023
Faculty Signature:
Grade:

Objective(s):

- To study Large Language Models (LLM).
- To build an efficient model using Generative AI based LLMs.

Outcome:

Students will be able to understand how to perform a Natural Language processing task using Generative AI based LLMs

Problem Statement:

To implement simple PDF Document search using Open Source Generative AI model.

Background Study: Generative AI based Language Models (LLMs) utilize deep learning to generate human-like text. These models are highly versatile and find application in diverse areas such as creative writing, chatbots, language translation, and even document search. LLMs can effectively process and understand natural language, making them valuable tools for searching and summarizing vast collections of documents.

Question Bank:

1. What is a Large Language Model?
   A Large Language Model (LLM) is a type of artificial intelligence model designed to understand, generate, and manipulate human language. It uses deep learning techniques, often based on transformer architectures, to process and generate text, enabling it to perform tasks like language translation, text generation, question answering, and more.

2. What is a transformer pipeline?
   A transformer pipeline refers to a sequence of processing steps using a transformer-based model, like GPT-3.5, bard. to perform specific tasks. It involves input data being passed through the model to receive relevant outputs. For example, a text generation pipeline might

involve input preprocessing, passing the input through the model, and then post-processing the model's generated text.

3. What are vector databases and embeddings?
   Vector databases and embeddings are techniques used in machine learning to represent data and enable efficient processing. A vector database stores data points as vectors in a multi-dimensional space, making it easier to search and compare them. Embeddings are lower-dimensional representations of data, often learned from raw data, which capture semantic relationships. For instance, word embeddings represent words in a way that preserves semantic similarities.

4. How does similarity search work?
   Vector databases and embeddings are techniques used in machine learning to represent data and enable efficient processing. A vector database stores data points as vectors in a multi-dimensional space, making it easier to search and compare them. Embeddings are lower-dimensional representations of data, often learned from raw data, which capture semantic relationships. For instance, word embeddings represent words in a way that preserves semantic similarities.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 15

Problem Statement:

To implement simple PDF Document search using Open Source Generative AI model.

Install Dependencies:

In [ ]:
```
! pip install langchain pypdf faiss-cpu sentence-transformers
```

```
Requirement already satisfied: langchain in c:\users\mainp\appdata\local\programs\python\pytho
n311\lib\site-packages (0.0.344)
Requirement already satisfied: pypdf in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (3.17.1)
Requirement already satisfied: faiss-cpu in c:\users\mainp\appdata\local\programs\python\pytho
n311\lib\site-packages (1.7.4)
Requirement already satisfied: sentence-transformers in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (2.2.2)
Requirement already satisfied: PyYAML>=5.3 in c:\users\mainp\appdata\local\programs\python\pyt
hon311\lib\site-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in c:\users\mainp\appdata\local\programs\pyt
hon\python311\lib\site-packages (from langchain) (2.0.23)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from langchain) (3.9.1)
Requirement already satisfied: anyio<4.0 in c:\users\mainp\appdata\local\programs\python\pytho
n311\lib\site-packages (from langchain) (3.7.1)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in c:\users\mainp\appdata\local\pr
ograms\python\python311\lib\site-packages (from langchain) (0.6.3)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in c:\users\mainp\appdata\local\programs\p
ython\python311\lib\site-packages (from langchain) (1.33)
Requirement already satisfied: langchain-core<0.1,>=0.0.8 in c:\users\mainp\appdata\local\prog
rams\python\python311\lib\site-packages (from langchain) (0.0.8)
Requirement already satisfied: langsmith<0.1.0,>=0.0.63 in c:\users\mainp\appdata\local\progra
ms\python\python311\lib\site-packages (from langchain) (0.0.67)
Requirement already satisfied: numpy<2,>=1 in c:\users\mainp\appdata\local\programs\python\pyt
hon311\lib\site-packages (from langchain) (1.25.0)
Requirement already satisfied: pydantic<3,>=1 in c:\users\mainp\appdata\local\programs\python
\python311\lib\site-packages (from langchain) (2.5.2)
Requirement already satisfied: requests<3,>=2 in c:\users\mainp\appdata\local\programs\python
\python311\lib\site-packages (from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from langchain) (8.2.3)
Requirement already satisfied: transformers<5.0.0,>=4.6.0 in c:\users\mainp\appdata\local\prog
rams\python\python311\lib\site-packages (from sentence-transformers) (4.35.2)
Requirement already satisfied: tqdm in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (from sentence-transformers) (4.65.0)
Requirement already satisfied: torch>=1.6.0 in c:\users\mainp\appdata\local\programs\python\py
thon311\lib\site-packages (from sentence-transformers) (2.1.1+cu118)
Requirement already satisfied: torchvision in c:\users\mainp\appdata\local\programs\python\pyt
hon311\lib\site-packages (from sentence-transformers) (0.16.1+cu118)
Requirement already satisfied: scikit-learn in c:\users\mainp\appdata\local\programs\python\py
thon311\lib\site-packages (from sentence-transformers) (1.2.2)
Requirement already satisfied: scipy in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (from sentence-transformers) (1.10.1)
Requirement already satisfied: nltk in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (from sentence-transformers) (3.8.1)
Requirement already satisfied: sentencepiece in c:\users\mainp\appdata\local\programs\python\p
ython311\lib\site-packages (from sentence-transformers) (0.1.99)
Requirement already satisfied: huggingface-hub>=0.4.0 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from sentence-transformers) (0.19.4)
Requirement already satisfied: attrs>=17.3.0 in c:\users\mainp\appdata\local\programs\python\p
ython311\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\mainp\appdata\local\programs\py
thon\python311\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\mainp\appdata\local\programs\python
\python311\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.3)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\mainp\appdata\local\programs\pyth
on\python311\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\mainp\appdata\local\programs\pytho
n\python311\lib\site-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.3.1)
Requirement already satisfied: idna>=2.8 in c:\users\mainp\appdata\local\programs\python\pytho
n311\lib\site-packages (from anyio<4.0->langchain) (3.4)
Requirement already satisfied: sniffio>=1.1 in c:\users\mainp\appdata\local\programs\python\py
thon311\lib\site-packages (from anyio<4.0->langchain) (1.3.0)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in c:\users\mainp\appdata\local\prog
rams\python\python311\lib\site-packages (from dataclasses-json<0.7,>=0.5.7->langchain) (3.20.
```

```
1)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in c:\users\mainp\appdata\local\progra
ms\python\python311\lib\site-packages (from dataclasses-json<0.7,>=0.5.7->langchain) (0.9.0)
Requirement already satisfied: filelock in c:\users\mainp\appdata\local\programs\python\python
311\lib\site-packages (from huggingface-hub>=0.4.0->sentence-transformers) (3.12.2)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\mainp\appdata\local\programs\pytho
n\python311\lib\site-packages (from huggingface-hub>=0.4.0->sentence-transformers) (2023.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\mainp\appdata\local\prog
rams\python\python311\lib\site-packages (from huggingface-hub>=0.4.0->sentence-transformers)
(4.6.3)
Requirement already satisfied: packaging>=20.9 in c:\users\mainp\appdata\local\programs\python
\python311\lib\site-packages (from huggingface-hub>=0.4.0->sentence-transformers) (23.1)
Requirement already satisfied: jsonpointer>=1.9 in c:\users\mainp\appdata\local\programs\pytho
n\python311\lib\site-packages (from jsonpatch<2.0,>=1.33->langchain) (2.4)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from pydantic<3,>=1->langchain) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from pydantic<3,>=1->langchain) (2.14.5)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mainp\appdata\local\progra
ms\python\python311\lib\site-packages (from requests<3,>=2->langchain) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mainp\appdata\local\programs\pyt
hon\python311\lib\site-packages (from requests<3,>=2->langchain) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mainp\appdata\local\programs\pyt
hon\python311\lib\site-packages (from requests<3,>=2->langchain) (2023.11.17)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\mainp\appdata\local\programs\pytho
n\python311\lib\site-packages (from SQLAlchemy<3,>=1.4->langchain) (3.0.1)
Requirement already satisfied: sympy in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (from torch>=1.6.0->sentence-transformers) (1.12)
Requirement already satisfied: networkx in c:\users\mainp\appdata\local\programs\python\python
311\lib\site-packages (from torch>=1.6.0->sentence-transformers) (3.1)
Requirement already satisfied: jinja2 in c:\users\mainp\appdata\local\programs\python\python31
1\lib\site-packages (from torch>=1.6.0->sentence-transformers) (3.1.2)
Requirement already satisfied: colorama in c:\users\mainp\appdata\local\programs\python\python
311\lib\site-packages (from tqdm->sentence-transformers) (0.4.6)
Requirement already satisfied: regex!=2019.12.17 in c:\users\mainp\appdata\local\programs\pyth
on\python311\lib\site-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers) (2023.
10.3)
Requirement already satisfied: tokenizers<0.19,>=0.14 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers)
(0.15.0)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\mainp\appdata\local\programs\pyt
hon\python311\lib\site-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers) (0.4.
1)
Requirement already satisfied: click in c:\users\mainp\appdata\local\programs\python\python311
\lib\site-packages (from nltk->sentence-transformers) (8.1.7)
Requirement already satisfied: joblib in c:\users\mainp\appdata\local\programs\python\python31
1\lib\site-packages (from nltk->sentence-transformers) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mainp\appdata\local\programs\p
ython\python311\lib\site-packages (from scikit-learn->sentence-transformers) (3.1.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from torchvision->sentence-transformers) (9.5.0)
Requirement already satisfied: mypy-extensions>=0.3.0 in c:\users\mainp\appdata\local\programs
\python\python311\lib\site-packages (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.
5.7->langchain) (1.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mainp\appdata\local\programs\python
\python311\lib\site-packages (from jinja2->torch>=1.6.0->sentence-transformers) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in c:\users\mainp\appdata\local\programs\python\py
thon311\lib\site-packages (from sympy->torch>=1.6.0->sentence-transformers) (1.3.0)
```

Code:

In [ ]:

```python
# importing required libraries
import os
from langchain.document_loaders import PyPDFLoader
from langchain.embeddings.openai import OpenAIEmbeddings
```

```
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.chains.question_answering import load_qa_chain
from langchain import HuggingFaceHub
from langchain.embeddings import HuggingFaceEmbeddings
```

In [ ]:
```
# setting up the huggingfacehub api token
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "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" # this is a
```

In [ ]:
```
# loading the pdf document using pyPDF and the document loader
pdf_loader = PyPDFLoader("https://scholarworks.calstate.edu/downloads/vq27zt20r")
pdf_document = pdf_loader.load()
```

In [ ]:
```
# converting the pdf document to raw text
pdf_text = ''

for i, page in enumerate(pdf_document):
    page_text = page.page_content
    if(page_text != None):
        pdf_text += page_text
```

In [ ]:
```
# splitting the document into chunks using RecursiveTextSplitter
text_splitter = CharacterTextSplitter(
    separator="\n",
    chunk_size=800,
    chunk_overlap=200,
    length_function=len,

)

# splitting the document into chunks
chunks = text_splitter.split_text(pdf_text)
```

In [ ]:
```
# loading the openai embeddings
embeddings = HuggingFaceEmbeddings()
```

In [ ]:
```
# creating the vector store
document_search = FAISS.from_texts(chunks, embeddings)
```

In [ ]:
```
# loading the Flan-T5 XL model from the huggingface hub

model = HuggingFaceHub(repo_id="google/flan-t5-xl",
                       model_kwargs={"temperature": 1, "max_length": 1000000})
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\huggingface_hub\utils
\_deprecation.py:127: FutureWarning: '__init__' (from 'huggingface_hub.inference_api') is depr
ecated and will be removed from version '1.0'. `InferenceApi` client is deprecated in favor of
the more feature-complete `InferenceClient`. Check out this guide to learn how to convert your
script to use it: https://huggingface.co/docs/huggingface_hub/guides/inference#legacy-inferenc
eapi-client.
  warnings.warn(warning_message, FutureWarning)
```

In [ ]:
```
# loading the question answering chain
chain = load_qa_chain(model, chain_type="stuff")
```

In [ ]:
```
query = "What are the algorithms used in the paper?"
docs = document_search.similarity_search(query)
answer = chain.run(input_documents=docs, question=query)
print(answer)
```

```
DQN [11] and A2C [21]
```