CCS337 - Cognitive Science Laboratory LAB Manual Record

Information Technolgy (St. Peter's College of Engineering and Technology)

St. PETER'S

COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution)

Affiliated to Anna University| Approved by AICTE
Avadi, Chennai, Tamilnadu – 600 054

Phone:7358110159/56 Website: www.spcet.ac.in Email:spcet2008@gmail.com

DEPARTMENT OF INFORMATION TECHNOLOGY

CCS337 – COGNITIVE SCIENCE LABORATORY

RECORD NOTEBOOK

NAME	**:**
REG.NO	**:**
BRANCH	**:**
YEAR/SEM	**:**

2024-2025

St. PETER'S
COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution)

Affiliated to Anna University| Approved by AICTE
Avadi, Chennai, Tamilnadu – 600 054

Phone: 7358110159/56 Website: www.spcet.ac.in Email:spcet2008@gmail.com

DEPARTMENT OF INFORMATION TECHNOLOGY

Bonafide Certificate

NAME……..…….

YEAR……..**SEMESTER**……..…….....

BRANCH……..……..…….....

REGISTER NO. ……...…….....

Certified that this bonafide record work done by the above student of the _____ _____during the year 2024 – 2025.

Faculty-in-Charge **Head of the Department**

Submitted for the practical Examination held on_____at St. PETER'S COLLEGE OF ENGINEERING AND TECHNOLOGY

Internal Examiner **External Examiner**

St. PETER'S

COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution)

Affiliated to Anna University | Approved by AICTE

Avadi, Chennai, Tamilnadu – 600 054

INSTITUTION VISION

To emerge as an Institution of Excellence by providing High Quality Education in Engineering, Technology and Management to contribute for the economic as well as societal growth of our Nation.

INSTITUTION MISSION

- To impart strong fundamental and Value-Based Academic knowledge in various Engineering, Technology and Management disciplines to nurture creativity.
- To promote innovative Research and Development activities by collaborating with Industries, R&D organizations and other statutory bodies.
- To provide conducive learning environment and training so as to empower the students with dynamic skill development for employability.
- To foster Entrepreneurial spirit amongst the students for making a positive impact on remark able community development.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

To emerge as a center of academic excellence to meet the industrial needs of the competitive world with IT technocrats and researchers for the social and economic growth of the country in the areaof Information Technology

MISSION

- To provide quality education to the students to attain new heights in IT industry and research
- To create employable students at national/international level by training them with adequate skills
- To produce good citizens with high personal and professional ethics to serve both the IT industry and society.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs):

Graduates will be able to

- Demonstrate technical competence with analytical and critical thinking to understand and meet the diversified requirements of industry, academia and research.
- Exhibit technical leadership, team skills and entrepreneurship skills to provide business solutionsto real world problems.
- Work in multi-disciplinary industries with social and environmental responsibility, work ethics and adaptability to address complex engineering and social problems
- Pursue lifelong learning, use cutting edge technologies and involve in applied research to design Optimal solutions.

PROGRAM OUTCOMES (POs):

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities withan understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need B.TECH. INFORMATION TECHNOLOGY 2 for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OBJECTIVES (PSOs)

To ensure graduates

- Have proficiency in programming skills to design, develop and apply appropriate techniques, to solve complex engineering problems.
- Have knowledge to build, automate and manage business solutions using cutting edge technologies.
- Have excitement towards research in applied computer technologies.

CCS337 – Cognitive Science Laboratory

COURSE OUTCOMES:

CO1: Understand the underlying theory behind cognition.

CO2: Connect to the cognition elements computationally.

CO3: Implement mathematical functions through WebPPL.

CO4: Develop applications using cognitive inference model.

CO5: Develop applications using cognitive learning model.

CO – PO & PSO's MAPPING:

COs	PO's												PSO's		
	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12	PSO-1	PSO-2	PSO-3
CO- 1	3	1	3	2	2	-	-	-	1	1	2	2	1	2	2
CO-2	2	2	1	1	2	-	-	-	3	2	3	1	2	3	2
CO-3	1	3	1	3	3	-	-	-	1	3	1	3	3	1	2
CO-4	2	1	1	2	2	-	-	-	1	2	3	1	3	3	1
CO-5	1	2	3	2	2	-	-	-	1	2	2	2	2	2	1
Avg	1.8	1.8	1.8	2	2.4	-	-	-	1.4	2	2.2	1.8	2.2	2.2	1.6

1 - low, 2 - medium, 3 - high, "-"- no correlation

CCS337 – Cognitive Science Laboratory

COURSE OBJECTIVES:

- To know the theoretical background of cognition.
- To understand the link between cognition and computational intelligence.
- To explore probabilistic programming language.
- To study the computational inference models of cognition.
- To study the computational learning models of cognition.

LIST OF EXPERIMENTS:

1. Demonstration of Mathematical functions using WebPPL.
2. Implementation of reasoning algorithms.
3. Developing an Application system using generative model.
4. Developing an Application using conditional inference learning model.
5. Application development using hierarchical model.
6. Application development using Mixture model.

TABLE OF CONTENTS

EX.NO:1

DEMONSTRATION OF MATHEMATICAL FUNCTIONS USING WEBPPL

Aim:

To demonstrate the mathematical functions using WebPPL.

Algorithm:

Initialize Basic Arithmetic Operations:

- Define variables for addition, subtraction, multiplication, and division.
- Perform the operations and store the results in variables.

Apply Advanced Mathematical Functions:

- Use built-in WebPPL Math functions to compute exponentiation, square roots, logarithms, and trigonometric values.
- Store the results in appropriate variables.

Generate a Random Number:

- Use the Math.random() function to generate a random number between 0 and 1.

Simulate Probabilistic Behavior:

- Use the flip function in WebPPL to simulate a fair coin toss (with a 0.5 probability for heads or tails).

Display the Results:

- Use the display() function to print the results of all computations and probabilistic outcomes.

Program:

```
// 1.Basic Arithmetic Operations
var add = 5 + 3;
var subtract = 10 - 6;
var multiply = 4 * 7;
var divide = 20 / 5;

display("Addition (5 + 3): " + add);
display("Subtraction (10 - 6): " + subtract);
display("Multiplication (4 * 7): " + multiply);
display("Division (20 / 5): " + divide);

// 2. Exponentiation
var power = Math.pow(2, 3); // 2^3
display("2 to the power of 3: " + power);

// 3. Square Root
var sqrt = Math.sqrt(16); // √16
display("Square root of 16: " + sqrt);

// 4. Trigonometric Functions
var sinVal = Math.sin(Math.PI / 2); // Sin(90 degrees)
var cosVal = Math.cos(0);           // Cos(0 degrees)
display("Sin(90 degrees): " + sinVal);
display("Cos(0 degrees): " + cosVal);

// 5. Logarithm
var logVal = Math.log(10); // Natural log of 10
display("Natural Log of 10: " + logVal);

// 6. Random Numbers
var randomNumber = Math.random(); // Random number between 0 and 1
display("Random Number: " + randomNumber);

// 7. Probability Distribution Example
var flipCoin = flip(0.5); // Simulates a fair coin toss
display("Coin Flip Result (1 = Heads, 0 = Tails): " + flipCoin)
```

Output:

```
Addition (5 + 3): 8
Subtraction (10 - 6): 4
Multiplication (4 * 7): 28
Division (20 / 5): 4
2 to the power of 3: 8
Square root of 16: 4
Sin(90 degrees): 1
Cos(0 degrees): 1
Natural Log of 10: 2.302585092994046
Random Number: 0.8278774263605473
Coin Flip Result (1 = Heads, 0 = Tails): false
```

Result:

Thus the demonstration of mathematical functions using WebPPL had been successfully implemented and the output is also verified.

EX.NO:2

IMPLEMENTATION OF REASONING ALGORITHMS

Aim:

To Implement the reasoning algorithms.

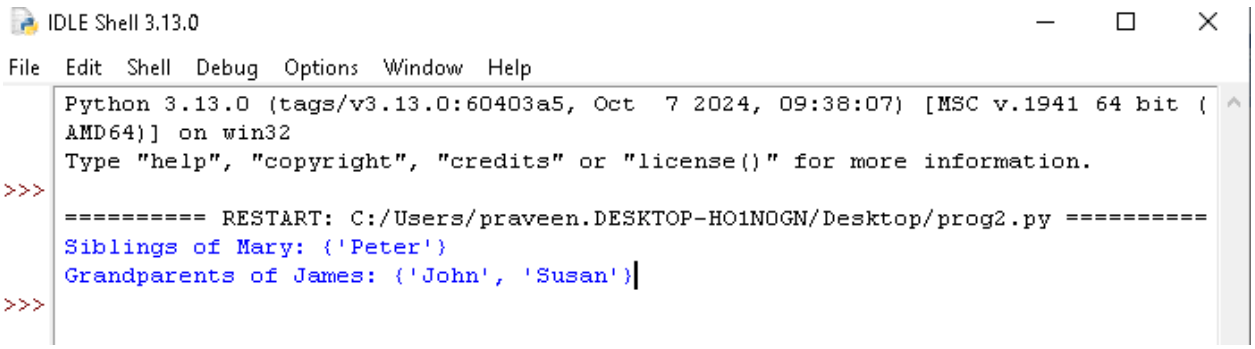
Algorithm:

1. Define Facts:
2. Store all parent-child relationships in a dictionary using tuples.
3. Key: Relationship type (e.g., "parent").
4. Value: A list of tuples representing relationships.
5. Create the find_siblings Function:
6. Input: The name of a person.
7. Logic:
8. Iterate through all parent-child pairs in the family dictionary.
9. Check for children with the same parent but different names.
10. Collect such children as siblings.
11. Output: A set of sibling names.
12. Create the find_grandparents Function:
13. Input: The name of a person.
14. Logic:
15. Iterate through all parent-child pairs to find the person's parent.
16. Use the parent’s name to find their parent (grandparent).
17. Collect such grandparents.
18. Output: A set of grandparent names.
19. Process the Queries:
20. Query for a specific person’s siblings by calling the find_siblings function.
21. Query for a specific person’s grandparents by calling the find_grandparents function.
22. Display Results:
23. Print the results of the queries for siblings and grandparents.

Program:

```
# Demonstrating reasoning with family relationships

# Facts
family = {
    "parent": [
        ("John", "Mary"),
        ("John", "Peter"),
        ("Susan", "Mary"),
        ("Susan", "Peter"),
        ("Mary", "James"),
        ("Mary", "Sophia"),
    ]
}

# Function to find siblings
def find_siblings(person):
    siblings = set()
    for p1, child1 in family["parent"]:
        for p2, child2 in family["parent"]:
            if p1 == p2 and child1 != child2 and child1 == person:
                siblings.add(child2)
    return siblings

# Function to find grandparents
def find_grandparents(person):
    grandparents = set()
    for grandparent, parent in family["parent"]:
        for p, child in family["parent"]:
            if parent == p and child == person:
                grandparents.add(grandparent)
    return grandparents

# Queries and Reasoning
query1 = "Mary"
siblings_of_query1 = find_siblings(query1)
print(f"Siblings of {query1}: {siblings_of_query1}")

query2 = "James"
grandparents_of_query2 = find_grandparents(query2)
print(f"Grandparents of {query2}: {grandparents_of_query2}")
```

Output:



Result:

Thus the Implementation of reasoning algorithms had been successfully implemented and the output is also verified.

EX.NO:3

DEVELOPING AN APPLICATION SYSTEM USING GENERATIVE MODEL

Aim:

To develop an application system using generative model.

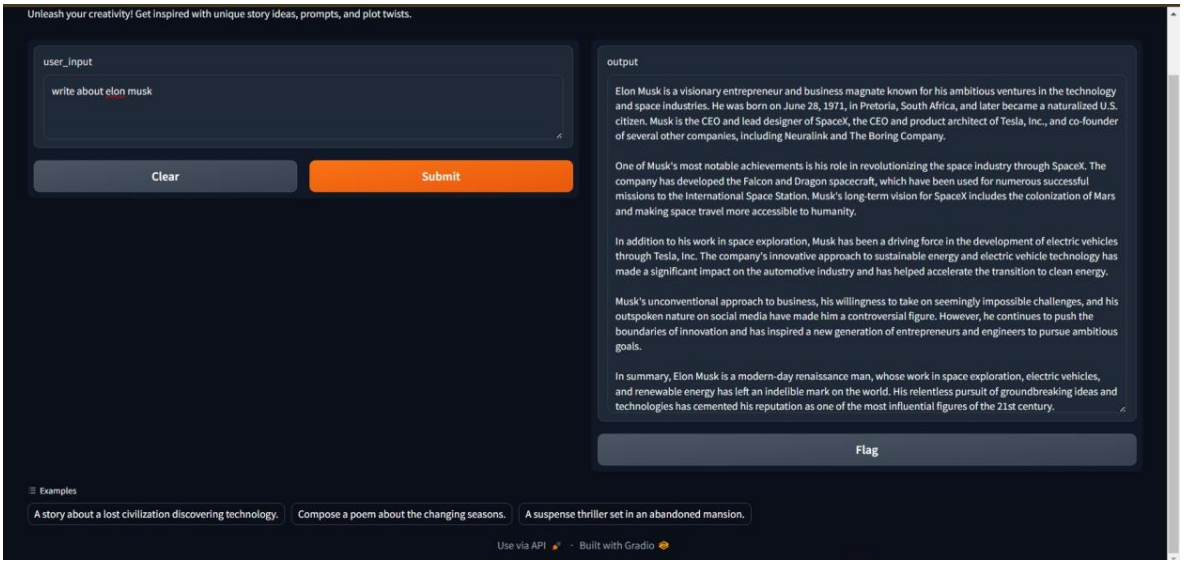
Algorithm:

1. Import Libraries: Load required libraries (g4f and gradio).
2. Set Up Client: Initialize the chat client to handle input and output.
3. Define Function: Create a function to process user input and generate responses.
4. Build Interface: Use Gradio to create a user-friendly web interface.
5. Run Program: Launch the interface for interaction.

Program:

```
from g4f.client import Client
import gradio as gr

client= Client()

def generate_writing_prompt(user_input):
    response=client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": user_input}],
    )
    returnresponse.choices[0].message.content

interface=gr.Interface(
    fn=generate_writing_prompt,
    inputs=gr.Textbox(lines=3, placeholder="Enter a genre, tone, or initial plot point..."),
    outputs="text",
    title="Creative Writing Assistant ",
    description="Unleash your creativity! Get inspired with unique story ideas, prompts, and plot
twists.",
    theme="huggingface",
    examples=[
        ["A story about a lost civilization discovering technology."],
        ["Compose a poem about the changing seasons."],
        ["A suspense thriller set in an abandoned mansion."],
    ]
)

# Launch the interface
If __name__=="__main__":
    interface.launch()
```

Output:



Result:

Thus the development of an application system using generative system had been successfully implemented and the output is also verified.

EX.NO:4

DEVELOPING AN APPLICATION USING CONDITIONAL INFERENCE LEARNING MODEL

Aim:

To Develop an application using conditional inference learning model.

Algorithm:

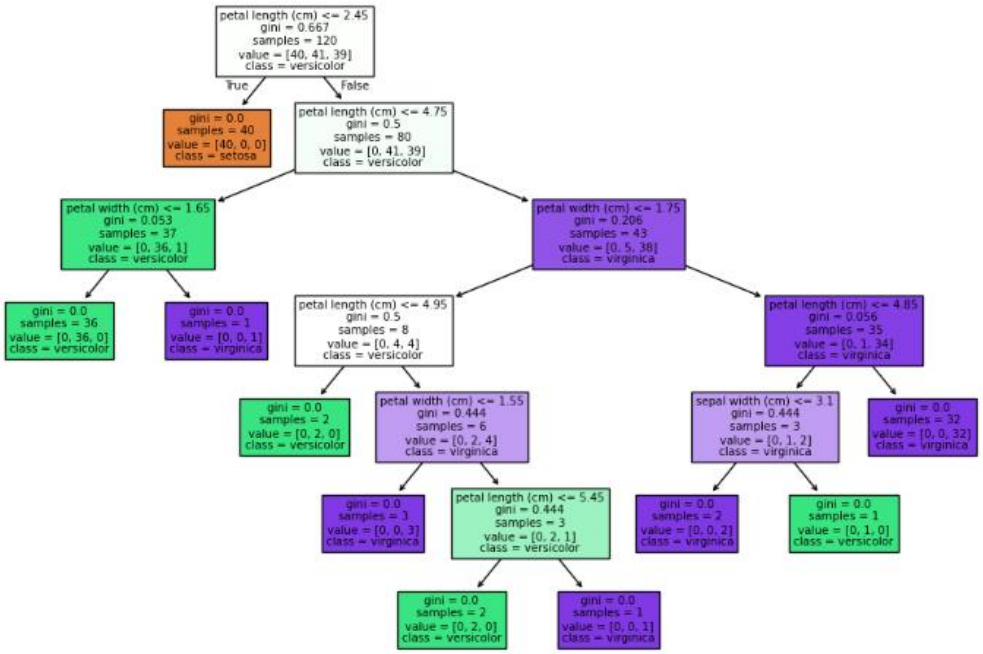
1. Input: Training dataset with features and target labels.
2. Initialization: Start with the entire dataset.
3. Splitting: At each step, split the dataset based on the feature that best separates the data according to some criteria (e.g., Gini impurity, entropy).
4. Recursion: Recursively split the data until the stopping criteria are met (e.g., a leaf node with a pure class or a maximum tree depth).
5. Output: A trained model (decision tree) that can predict class labels based on input features.

Program:

```
# Import necessary libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
import pandas as pd

# Load a sample dataset (Iris dataset in this case)
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Decision Tree Classifier (a basic conditional inference model)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Display the decision tree structure (optional)
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=data.feature_names, class_names=data.target_names)
plt.show()
```

Output:

```
Model Accuracy: 100.00%
```



Result:

Thus the Development of an application using conditional inference learning model had been successfully implemented and the output is also verified.

EX.NO:5

APPLICATION DEVELOPMENT USING HIERARCHICAL MODEL

Aim:

To Develop an application using hierarchical model.

Algorithm:

1. Company name, department names, employee names and positions, and sub-department structure.
2. Initialization
3. Create the company, departments, and employees.
4. Data Addition
5. Add employees to departments.
6. Add sub-departments to departments.
7. Display Hierarchy
8. Print the company name.
9. Print the departments, their employees, and any sub-departments.
10. Output
11. Display the hierarchical structure of the company, showing relationships between departments and employees.

Program:

```
# Define a class to represent an employee
class Employee:
    def __init__(self, name, position):
        self.name = name
        self.position = position

    def __repr__(self):
        return f"Employee(name={self.name}, position={self.position})"


# Define a class to represent a department
class Department:
    def __init__(self, name):
        self.name = name
        self.employees = []  # List of employees in this department
        self.sub_departments = []  # List of sub-departments under this department

    def add_employee(self, employee):
        self.employees.append(employee)

    def add_sub_department(self, department):
        self.sub_departments.append(department)

    def __repr__(self):
        return f"Department(name={self.name}, employees={len(self.employees)},
sub_departments={len(self.sub_departments)})"


# Define a class to represent a company
class Company:
    def __init__(self, name):
        self.name = name
        self.departments = []

    def add_department(self, department):
        self.departments.append(department)

    def display_hierarchy(self):
        print(f"Company: {self.name}")
        self._display_departments(self.departments, indent=2)
```

```
    def _display_departments(self, departments, indent):
        for department in departments:
            print(" " * indent + f"Department: {department.name}")
            for employee in department.employees:
                print(" " * (indent + 2) + f"Employee: {employee.name}, Position:
{employee.position}")
            if department.sub_departments:
                self._display_departments(department.sub_departments, indent + 2)


# Example usage of the Hierarchical Model

# Create a company
my_company = Company("Tech Innovators")

# Create departments
sales_department = Department("Sales")
hr_department = Department("Human Resources")

# Add employees to departments
sales_department.add_employee(Employee("Alice", "Sales Manager"))
sales_department.add_employee(Employee("Bob", "Sales Associate"))
hr_department.add_employee(Employee("Charlie", "HR Manager"))

# Create sub-department and add to a parent department
recruitment_department = Department("Recruitment")
hr_department.add_sub_department(recruitment_department)
recruitment_department.add_employee(Employee("David", "Recruiter"))

# Add departments to the company
my_company.add_department(sales_department)
my_company.add_department(hr_department)

# Display the hierarchy of the company
my_company.display_hierarchy()
```

Output:

```
Company: Tech Innovators
  Department: Sales
    Employee: Alice, Position: Sales Manager
    Employee: Bob, Position: Sales Associate
  Department: Human Resources
    Employee: Charlie, Position: HR Manager
    Department: Recruitment
      Employee: David, Position: Recruiter
```

Result:

Thus the Application development using hierarchical model had been successfullyimplemented and the output is also verified.

EX.NO:6

APPLICATION DEVELOPMENT USING MIXTURE MODEL

Aim:

To Develop an application using mixture model.

Algorithm:

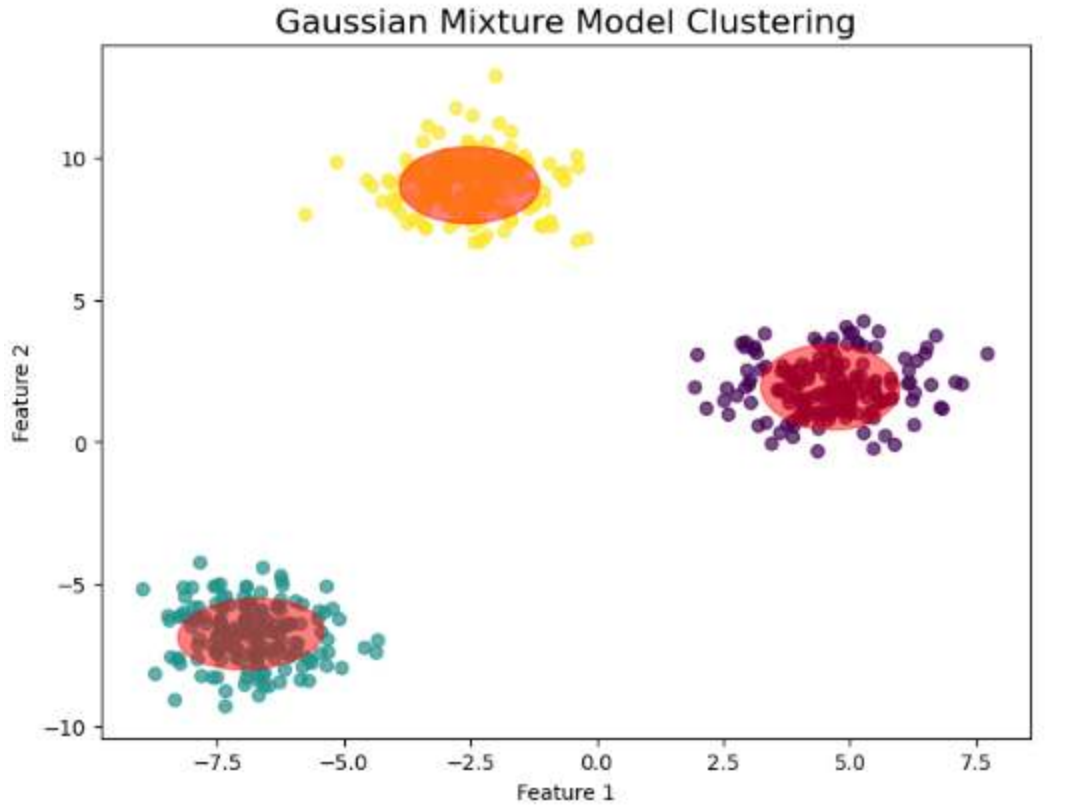
1. Generate Data
2. Use make_blobs from sklearn.datasets to generate synthetic data points with multiple clusters.
3. Fit Gaussian Mixture Model
4. Fit a Gaussian Mixture model to the generated data to identify the underlying Gaussian components in the data.
5. Predict Cluster Labels
6. Use the trained GMM to predict the cluster labels of the data points.
7. Visualize Data and GMM Components
8. Plot the data points with cluster labels using matplotlib.
9. For each Gaussian component, compute the covariance, eigenvalues, and eigenvectors, and plot an ellipse representing the component.
10. Display the Plot
11. Display the scatter plot with ellipses showing the Gaussian distributions for each component.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs
```

```
# Step 1: Generate synthetic data (3 clusters)
X, _ = make_blobs(n_samples=500, centers=3, random_state=42)

# Step 2: Fit the Gaussian Mixture Model (GMM) to the data

gmm = GaussianMixture(n_components=3, covariance_type='full', random_state=42)
gmm.fit(X)

# Step 3: Predict the labels (clusters)
labels = gmm.predict(X)

# Step 4: Visualize the data and the predicted clusters
plt.figure(figsize=(8, 6))

# Scatter plot of the data points
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', marker='o', alpha=0.7)

# Plot the Gaussian components (ellipses)
ax = plt.gca()
for mean, covar in zip(gmm.means_, gmm.covariances_):
    v, w = np.linalg.eigh(covar)
    v = 2.0 * np.sqrt(2.0) * np.sqrt(v)  # Scale the ellipse by a factor of 2
    u = w[0] / np.linalg.norm(w[0])
    angle = np.arctan(u[1] / u[0])
    angle = 180.0 * angle / np.pi
    angle = angle + 90

    # Create the ellipse patch with keyword arguments for rotation
    ell = plt.matplotlib.patches.Ellipse(mean, v[0], v[1], angle=180.0 + angle, color='red',
          alpha=0.5)
    ax.add_patch(ell)

# Labels and title
plt.title('Gaussian Mixture Model Clustering', fontsize=16)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

plt.show()
```

Output:



Result:

Thus the Development of an application using mixture model had been successfully implemented and the output is also verified.