DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

LABORATORY MANUAL

[R20A0580]

B. TECH CSE
(III YEAR–IISEM)

R20 REGULATION
(2023-24)

Name :	
Roll no:	
Section:	
Year :	

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AutonomousInstitution– UGC,Govt.ofIndia)
Recognizedunder2(f)and12(B) ofUGCACT1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA &NAAC–'A' Grade-ISO9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100,TelanganaState,India

INDEX

S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Vision

To establish a pedestal for the integral innovation, team spirit, originality and competence in the students, expose them to face the global challenges and become technology leaders of Indian vision of modern society.

Mission

- To become a model institution in the fields of Engineering, Technology and Management.
- To impart holistic education to the students to render them as industry ready engineers.
- To ensure synchronization of MRCET ideologies with challenging demands of International Pioneering Organizations.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1–ANALYTICALSKILLS

- ❖ To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2–TECHNICALSKILLS

- ❖ Tofacilitatethegraduateswiththetechnicalskillsthatpreparethemforimmediateemployment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3– SOFTSKILLS

- ❖ To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self confidence by communicating effectively, having a positive attitude ,get involved in team-work, being a leader, managing their career and their life.

PEO4–PROFESSIONALETHICS

- ❖ To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting them to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B.Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1. FundamentalsandcriticalknowledgeoftheComputerSystem:-AbletoUnderstand the working principles of the computer System and its components ,Apply the knowledge to build, asses, and analyze the software and hardware aspects of it.

2. The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify their search gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

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 anddesignsystemcomponentsorprocessesthatmeetthespecifiedneedswithappropriateconsideration 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 with an 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 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.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY

1. Course Objectives:

1. Familiarity with the Prolog programming environment.
2. To introduce students to the basic concepts and techniques of Machine Learning.
3. To implement classification and clustering methods.
4. To become familiar with various supervised and unsupervised learning Algorithms.
5. Learning basic concepts of Prolog through illustrative examples and small exercises

& Understanding list data structure in Prolog.

2. Course outcomes:

1. Apply various AI search algorithms (uninformed, informed, heuristic, constraint satisfaction,)
2. Understand the fundamentals of knowledge representation, inference using AI tools..
3. Solve the problems using various machine learning techniques
4. Design application using machine learning techniques

3. Introduction about lab

Minimum System requirements:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor.
- Disk space: 1 GB.
- Operating systems: Windows* 7 or later, mac OS, and Linux.
- Python* versions: 2.7.X, 3.6.X., 3.8.X

About lab:

Python is a general purpose, high-level programming language; other high Level languages you might have heard of C++, PHP, Java and Python. Virtually all modern programming languages make us of an Integrated Development Environment (IDE), which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle).

Many modern languages use both processes. They are first compiled into a lower level language, called byte code, and then interpreted by a program called a virtual machine. Python uses both processes, but because of the way programmers interact with it, it is usually considered an interpreted language Practical aspects are the key to understanding and conceptual visualization Of theoretical aspects covered in the laboratory

4. Guidelines to students

A. Standard operating procedure

a) Explanation on today's experiment by the concerned faculty using PPT covering the following aspects:

1) Name of the experiment

2) Aim

b) Writing the python programs by the students c) Commands for executing programs

Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

a) Name of the experiment b) Aim

c) Writing the program

d) Viva-Voce Questions and Answers

e) Errors observed (if any) during compilation/execution f) Signature of the Faculty

B. Guide Lines to Students in Lab

Disciplinary to be maintained by the students in the Lab

- Students are required to carry their lab observation book and record book with completed experiments while entering the lab.
- Students must use the equipment with care. Any damage is caused student is punishable
- Students are not allowed to use their cell phones/pen drives/ CDs in labs.
- Students need to be maintain proper dress code along with ID Card
- Students are supposed to occupy the computers allotted to them and are not supposed to talk or make noise in the lab. Students, after completion of each experiment they need to be updated in observation notes and same to be updated in the record.
- Lab records need to be submitted after completion of experiment and get it corrected with the concerned lab faculty.
- If a student is absent for any lab, they need to be completed the same experiment in the free time before attending next lab.

Instructions to maintain the record

 Before start of the first lab they have to buy the record and bring the record to the lab.

 Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.

 In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.

 If record is not submitted in time or record is not written properly, the evaluation marks (5M) will be deducted.

C. General laboratory instructions

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session. b. Laboratory Record updated up to the last session experiments and other utensils
(if any) needed in the lab. c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

INDEX

Week 1: Write a program to Illustrate Different Set Operations?

Program:

```
# Program to perform different set operations like in mathematics
# define three sets

E = {0, 2, 4, 6, 8};
N = {1, 2, 3, 4, 5};

# set union
print("Union of E and N is",E | N)
# set intersection
print("Intersection of E and N is",E & N)

# set difference
print("Difference of E and N is",E - N)

# set symmetric difference
print("Symmetric difference of E and N is",E ^ N)
```

Output:

Viva Questions:

1. Define Artificial intelligence
2. What is list in python?
3. Difference between informed and uninformed search.
4. What is an agent?

Faculty Signature

Week2: Implementation of DFS for water jug

Program:

```
# This function is used to initialize the
# dictionary elements with a default value. from
collections import defaultdict

# jug1 and jug2 contain the value
# for max capacity in respective jugs
# and aim is the amount of water to be measured.
jug1, jug2, aim = 4, 3, 2

# Initialize dictionary with
# default value as false.
visited = defaultdict(lambda: False)

# Recursive function which prints the
# intermediate steps to reach the final
# solution and return boolean value
# (True if solution is possible, otherwise False).
# amt1 and amt2 are the amount of waterpresent
# in both jugs at a certain point of time.
def waterJugSolver(amt1, amt2):

# Checks for our goaland
# returns true ifachieved.
if (amt1 == aim and amt2 == 0) or (amt2 == aim and amt1 == 0):
            print(amt1, amt2)
return True

# Checks if we have already visited the
# combination or not. If not, then it proceeds further.
if visited[(amt1, amt2)] == False:
            print(amt1, amt2)

# Changes the boolean value of
# the combination as it is visited. visited[(amt1, amt2)] = True

# Check for all the 6 possibilities and

# see if a solution is found in any one of them.
Return (waterJugSolver(0, amt2) or
waterJugSolver(amt1, 0) or waterJugSolver(jug1, amt2) or waterJugSolver(amt1, jug2) or
waterJugSolver(amt1 + min(amt2, (jug1-amt1)), amt2 - min(amt2, (jug1-amt1))) or
waterJugSolver(amt1 - min(amt1, (jug2-amt2)), amt2 + min(amt1, (jug2-amt2))))
# Return False if the combination is
# already visited to avoid repetition otherwise
# recursion will enter an infinite loop.
else:
return False
print("Steps: ")

# Call the function and pass the
# initial amount of water present in both jugs.
waterJugSolver(0, 0)
```

Output:

Viva Questions:

1. **Difference between DFS and BFS**
2. **What is list indexing and slicing with an example**

Faculty Signature

Week3: Implementation of BFS for tic-tac-toe problem

```
import os import time

board = [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ']
player = 1

########win Flags##########
Win = 1
Draw = -1
Running = 0
Stop = 1
###########################
Game = Running
Mark = 'X'

#This Function Draws Game Board def
DrawBoard():
print(" %c | %c | %c " % (board[1],board[2],board[3]))
print("___|___|___")
print(" %c | %c | %c " % (board[4],board[5],board[6]))
print("___|___|___")
print(" %c | %c | %c " % (board[7],board[8],board[9]))
print(" | | ")

#This Function Checks position is empty or not def
CheckPosition(x):
if(board[x] == ' '):
        return True
else:
        return False

#This Function Checks player has won ornot def
CheckWin():
global Game
#Horizontal winning condition
if(board[1] == board[2] and board[2] == board[3] and board[1] != ' '):
        Game = Win
elif(board[4] == board[5] and board[5] == board[6] and board[4] != ' '):
        Game = Win
elif(board[7] == board[8] and board[8] == board[9] and board[7] != ' '):
        Game = Win
#Vertical Winning Condition
elif(board[1] == board[4] and board[4] == board[7] and board[1] != ' '):
        Game = Win
elif(board[2] == board[5] and board[5] == board[8] and board[2] != ' '):
        Game = Win
elif(board[3] == board[6] and board[6] == board[9] and board[3] != ' '):
        Game=Win


#Diagonal Winning Condition
elif(board[1] == board[5] and board[5] == board[9] and board[5] != ' '): Game = Win
elif(board[3] == board[5] and board[5] == board[7] and board[5] != ' '): Game=Win
#Match Tie or Draw Condition
elif(board[1]!=' ' and board[2]!=' ' and board[3]!=' ' and board[4]!=' ' and board[5]!=' ' and
board[6]!=' ' and board[7]!=' ' and board[8]!=' ' and board[9]!=' '): Game=Draw
else: Game=Running
print("Tic-Tac-Toe Game Designed By Sourabh Somani")
print("Player 1 [X] --- Player 2 [O]\n")
print()
print()
print("Please Wait...")
time.sleep(3)
while(Game == Running): os.system('cls') DrawBoard()
if(player % 2 != 0):
print("Player 1'schance")
Mark = 'X'
else:
print("Player 2's chance")
Mark = 'O'
choice = int(input("Enter the position between [1-9] where you want to mark : "))
if(CheckPosition(choice)): board[choice] = Mark player+=1
CheckWin()

os.system('cls') DrawBoard() if(Game==Draw):
print("Game Draw")
elif(Game==Win):
player-=1 if(player%2!=0):
print("Player 1 Won")
else:
print("Player 2 Won")
```

Output:

Viva Questions:

1. **What is BFS?**
2. **What is the difference between a Mutable data type and an Immutable data type?**
3. **What is A* algorithm?**

Faculty Signature

Week4: Solve 8-puzzle problem using best first search

```
import sys
import numpy as np

class Node:
    def __init__(self, state, parent, action):
        self.state = state
        self.parent = parent
        self.action = action

class StackFrontier:
    def __init__(self):
        self.frontier = []

    def add(self, node):
        self.frontier.append(node)

    def contains_state(self, state):
        return any((node.state[0] == state[0]).all() for node in self.frontier)

    def empty(self):
        return len(self.frontier) == 0

    def remove(self):
        if self.empty():
            raise Exception("Empty Frontier")
        else:
            node = self.frontier[-1]
            self.frontier = self.frontier[:-1]
            return node


class QueueFrontier(StackFrontier):
    def remove(self):
        if self.empty():
            raise Exception("Empty Frontier")
        else:
            node = self.frontier[0]
            self.frontier = self.frontier[1:]
            return node


class Puzzle:
    def __init__(self, start, startIndex, goal, goalIndex):
        self.start = [start, startIndex]
        self.goal = [goal, goalIndex]
        self.solution = None

    def neighbors(self, state):
        mat, (row, col) = state
        results = []

        if row > 0:
            mat1 = np.copy(mat)
            mat1[row][col] = mat1[row - 1][col]
            mat1[row - 1][col] = 0
            results.append(('up', [mat1, (row - 1, col)]))
        if col > 0:
            mat1 = np.copy(mat)
```

```
                mat1[row][col] = mat1[row][col - 1]
                mat1[row][col - 1] = 0
                results.append(('left', [mat1, (row, col - 1)]))
            if row < 2:
                mat1 = np.copy(mat)
                mat1[row][col] = mat1[row + 1][col]
                mat1[row + 1][col] = 0
                results.append(('down', [mat1, (row + 1, col)]))
            if col < 2:
                mat1 = np.copy(mat)
                mat1[row][col] = mat1[row][col + 1]
                mat1[row][col + 1] = 0
                results.append(('right', [mat1, (row, col + 1)]))

            return results

        def print(self):
            solution = self.solution if self.solution is not None else None
            print("Start State:\n", self.start[0], "\n")
            print("Goal State:\n",  self.goal[0], "\n")
            print("\nStates Explored: ", self.num_explored, "\n")
            print("Solution:\n ")
            for action, cell in zip(solution[0], solution[1]):
                print("action: ", action, "\n", cell[0], "\n")
            print("Goal Reached!!")

        def does_not_contain_state(self, state):
            for st in self.explored:
                if (st[0] == state[0]).all():
                    return False
            return True

        def solve(self):
            self.num_explored = 0

            start = Node(state=self.start, parent=None, action=None)
            frontier = QueueFrontier()
            frontier.add(start)

            self.explored = []

            while True:
                if frontier.empty():
                    raise Exception("No solution")

                node = frontier.remove()
                self.num_explored += 1

                if (node.state[0] == self.goal[0]).all():
                    actions = []
                    cells = []
                    while node.parent is not None:
                        actions.append(node.action)
                        cells.append(node.state)
                        node = node.parent
                    actions.reverse()
                    cells.reverse()
                    self.solution = (actions,  cells)
                    return
                self.explored.append(node.state)
                for action, state in self.neighbors(node.state):
                    if not frontier.contains_state(state) and self.does_not_contain_state(state):
                        child = Node(state=state, parent=node, action=action)
                        frontier.add(child)


start = np.array([[1, 2, 3], [8, 0, 4], [7, 6, 5]])
goal = np.array([[2, 8, 1], [0, 4, 3], [7, 6, 5]])


startIndex = (1, 1)
goalIndex = (1, 0)


p = Puzzle(start, startIndex, goal, goalIndex)
p.solve()
p.print()
```

Output:

```
Start State:
 [[1 2 3]
 [8 0 4]
 [7 6 5]]

Goal State:
 [[2 8 1]
 [0 4 3]
 [7 6 5]]
```

Viva Questions:

1. **What is Min Max Algorithm?**
2. **What are AND OR graphs?**

Faculty Signature

Week5: Write a program to solve 8 queens problem

```
# Taking number of queens as input from user
print ("Enter the number of queens")
N = int(input())
# here we create a chessboard
# NxN matrix with all elements set to 0
board = [[0]*N for _ in range(N)]
def attack(i, j):
   #checking vertically and horizontally
   for k in range(0,N):
      if board[i][k]==1 or board[k][j]==1:
         return True
   #checking diagonally
   for k in range(0,N):
      for l in range(0,N):
         if (k+l==i+j) or (k-l==i-j):
            if board[k][l]==1:
               return True
   return False
def N_queens(n):
   if n==0:
      return True
   for i in range(0,N):
      for j in range(0,N):
         if (not(attack(i,j))) and (board[i][j]!=1):
            board[i][j] = 1
            if N_queens(n-1)==True:
               return True
            board[i][j] = 0
   return False
N_queens(N)
for i in board:
   print (i)
```

Output:

Viva Questions:
1. Difference between forward chaining and backward chaining
2.Describe Alpha beta pruning

Faculty Signature

Week6: Write a program to implement Hill Climbing Algorithm

Program:

```
import random

def randomSolution(tsp): cities

=list(range(len(tsp))) solution = []

for i in range(len(tsp)):

randomCity = cities[random.randint(0, len(cities) - 1)]

solution.append(randomCity) cities.remove(randomCity)

return solution

def routeLength(tsp, solution):

routeLength = 0

for i in range(len(solution)):

routeLength += tsp[solution[i - 1]][solution[i]]

return routeLength

def getNeighbours(solution):

neighbours = []

for i in range(len(solution)):

for j in range(i + 1, len(solution)): neighbour =

solution.copy() neighbour[i] = solution[j]

neighbour[j] = solution[i]

            neighbours.append(neighbour)

return neighbours

def getBestNeighbour(tsp, neighbours): bestRouteLength =

routeLength(tsp, neighbours[0]) bestNeighbour = neighbours[0]

for neighbour in neighbours:

currentRouteLength = routeLength(tsp, neighbour)

if currentRouteLength < bestRouteLength:

bestRouteLength = currentRouteLength bestNeighbour =

neighbour

return bestNeighbour, bestRouteLength
```

```
    def hillClimbing(tsp):

currentSolution = randomSolution(tsp) currentRouteLength =

routeLength(tsp, currentSolution) neighbours =

getNeighbours(currentSolution)

bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

while bestNeighbourRouteLength < currentRouteLength:

currentSolution = bestNeighbour currentRouteLength =

bestNeighbourRouteLength neighbours =

getNeighbours(currentSolution)

bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

return currentSolution, currentRouteLength def main():

tsp = [

[0,  400,  500,  300],  [400,  0,

300, 500], [500, 300, 0, 400],

[300, 500, 400, 0]

]


print(hillClimbing(tsp))


if__name__== " main ":   _

main()
```

Output:

Viva Questions:

1. **Define hill climbing**
2. **What is stochastic search?**
3. **What are the types of hill climbing?**
4. **What is mean by global maxima and local maxima?**

Faculty Signature

Week7: Data Extraction, Wrangling

1. Loading different types of dataset in Python
2. Arranging the data

Go to Google Page and find www.kaggle.com .Select Datasets and find Titanic dataset , then download train.csv file ans save it to desktop.

1. Read a CSV file import pandas as
pd url='C:/Users/MRCET1/Desk
top/train.csv'
dataframe=pd.read_csv(url)
dataframe.head(5)

2. Write a CSV file import pandas as pd
marks_data=pd.DataFrame({'ID':{0:23,1:43,2:12,3:13,4:67,5:89},'NAME':{0:'Ram',1
:'Deep',2:'Ya sh',3:'Arjun',4:'Aditya',5:'Divya'},'Marks':{0:89,1:92,2:45,3:78,4:56,5:76},'Grade':{0:'
b',1:'a',2:'f',3:' c',4:'e',5:'c'}})

filename='C:/Users/MRCET1/Desktop/M
arksdata.xlsx' marks_data.to_excel(filename)

print('Data frame written to Excel')

3. Read an Excel File import pandas as pd

url='C:/Users/MRCET1/Desktop/train.csv.xls'

dataframe=pd.read_excel(url) dataframe.head(5)

4. Write an Excel file import pandas as pd

marks_data=pd.DataFrame({'ID':{0:23,1:43,2:12,3:13,4:67,5:89},'NAME':{0:'Ram',1
:'Deep',2:'Y

ash',3:'Arjun',4:'Aditya',5:'Divya'},'Marks':{0:89,1:92,2:45,3:78,4:56,5:76},'Grade':{0
:'b',1:'a',2:'f',

3:'c',4:'e',5:'c'}})
filename='C:/Users/MRCET1/Desktop/Marksdata.csv'
marks_data.to_csv(filename)

print('Data frame written to CSV');

Output:

Viva Questions

1. **What is Machine learning?**
2. **What is Numpy?**
3. **What is CSV?**
4. **What Are the Different Types of Machine Learning?**

Faculty Signature

WEEK-8
Data Visualization

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Downdload dataset and read it
csv_url = &#39;https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data&#39;
# using the attribute information as the column names col_names =
[&#39;Sepal_Length&#39;,&#39;Sepal_Width&#39;,&#39;Petal_Length&#39;,&#39;Peta l_Width
&#39;,&#39;Class&#39;]
iris = pd.read_csv(csv_url, names = col_names)
iris.head() iris[&quot;Class&quot;].value_counts()
# Line plots
import numpy as np
x = np.linspace(0,20,30)
y= x**2 plt.plot(x, y) plt.show()
# Line plot with grid
x = np.linspace(0,20,30)
y= x**2 plt.plot(x, y)
plt.xlabel(&#39;x-values&#39;) plt.ylabel(&#39;x^2- values&#39;)
plt.title(&#39;line plot&#39;) plt.grid(True) plt.show()

# Scatter Plot
iris.plot(kind=&quot;scatter&quot;, x=&quot;Sepal_Length&quot;,
y=&quot;Sepal_Width&quot;)
colours = {&#39;Iris-setosa&#39;:&#39;orange&#39;, &#39;Iris-
versicolor&#39;:&#39;lightgreen&#39;, &#39;Iris-
virginica&#39;:&#39;lightblue&#39;}
for i in range(len(iris[&#39;Sepal_Length&#39;])):
plt.scatter(iris[&#39;Petal_Length&#39;][i],iris[&#39;Petal_Width&#39;][i], color =
colours[iris[&#39;Class&#39;][i]])
plt.title(&#39;Iris&#39;) plt.xlabel(&#39;petal length&#39;)
plt.ylabel(&#39;petal width&#39;) plt.grid(True)
plt.show()
```

```
# We can also use the seaborn library to make a similar plot
sns.jointplot(x=&quot;Sepal_Length&quot;, y=&quot;Sepal_Width&quot;, data=iris,
size=5)
# Bar Graph
a= iris[&#39;Class&#39;].value_counts() species = a.index
count = a.values plt.bar(species,count,color =
&#39;lightgreen&#39;) plt.xlabel(&#39;species&#39;)
plt.ylabel(&#39;count&#39;)
plt.show()


# Box Plot length_width =
iris[[&#39;Petal_Length&#39;,&#39;Petal_Width&#39;,&#39;Sepal_Length&#39;,&#3
9;Sepal_Wi dth&#39;]] #excluding species
column length_width.boxplot() plt.xlabel(&#39;Flower
measurements&#39;) plt.ylabel(&#39;values&#39;)
plt.title(&quot;Iris dataset analysis&quot;)
# We can look at an individual feature in Seaborn through many different kinds of plots.
# Here&#39;s a boxplot
sns.boxplot(x=&quot;Class&quot;, y=&quot;Petal_Length&quot;,
palette=&quot;husl&quot;, data=iris)
#Histogram
import numpy as np
data_ = np.random.randn(1000) plt.hist(data_,bins =
40,color=&#39;gold&#39;) plt.grid(True)
plt.xlabel(&#39;points&#39;)
plt.title(&quot;Histogram&quot;) plt.show()
#Correlation Matrix correlation = iris.corr() fig ,ax = plt.subplots() k =
ax.imshow(correlation, cmap = &#39;magma_r&#39;)
ax.set_xticks(np.arange(len(correlation.columns)))
ax.set_yticks(np.arange(len(correlation.columns)))
ax.set_xticklabels(correlation.columns) ax.set_yticklabels(correlation.columns)
cbar = ax.figure.colorbar(k, ax=ax)
cbar.ax.set_ylabel(&#39;color bar&#39;, rotation=-90, va=&quot;bottom&quot;)

plt.setp(ax.get_xticklabels(), rotation=45,
ha=&quot;right&quot;,rotation_mode=&quot;anchor&quot;) for i in
range(len(correlation.columns)):
for j in range(len(correlation.columns)):
```

```
text = ax.text(j, i, np.around(correlation.iloc[i, j],
decimals=2),ha=&quot;center&quot;, va=&quot;center&quot;,
color=&quot;lightgreen&quot;)
plt.show()
#Piechart
a= iris[&#39;Class&#39;].value_counts()
species = a.index count = a.values
colors= [&#39;lightblue&#39;,&#39;lightgreen&#39;,&#39;gold&#39;]
explode = (0,0.2,0)
plt.pie(count, labels=species,shadow=True,
colors=colors,explode = explode, autopct=&#39;%1.1f%%&#39;)
plt.xlabel(&#39;species&#39;)
plt.axis(&#39;equal&#39;) plt.show()
sns.set_style(&#39;darkgrid&#39;)
sns.lineplot(data=iris.drop([&#39;Class&#39;], axis=1))
plt.show()
```

Output:

Viva Questions:

1. **List various data visualization tools for ML**
2. **What is meant by Data set?**
3. **What is pandas?**

Faculty Signature

Week9: Implementation of Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt def
estimate_coef(x, y):
    # number of observations/points n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients b_1 =
    SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

    def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m", marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x
    # plotting the regression line
     plt.plot(x, y_pred, color = "g")

    # putting labels plt.xlabel('x') plt.ylabel('y')

    # function to show plot plt.show()

    def main():
    # observations / data
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
        b = estimate_coef(x, y)

        print("Estimated coefficients:\nb_0 = {} \
                    \nb_1 = {}".format(b[0], b[1]))

        # plotting regression line
        plot_regression_line(x, y, b)

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

Output:

Viva Questions:

1. **What is regression in machine learning**
2. **What is meant by linear regression**
3. **Difference between linear and logistic regression**

Faculty Signature

Week10: Implementation of K-nearest Neighbor

```
import numpy as np import
pandas as pd
from sklearn.model_selection
import train_test_split
from sklearn.neighbors import
KNeighborsClassifier import
matplotlib.pyplot as plt import
seaborn as sns
df = pd.read_csv('data.csv')
y = df['diagnosis']
X = df.drop('diagnosis', axis=1)
X = X.drop('Unnamed: 32', axis=1)
X = X.drop('id', axis=1)

# Separating the dependent and independent variable
X_train, X_test, y_train, y_test = train_test_split( X, y,
    test_size=0.3, random_state=0)


# Splitting the data into training and testing data
K = []
training = []
test = []
scores = {}
for k in range(2, 21):
    clf = KNeighborsClassifier(n_neighbors=k)
    clf.fit(X_train, y_train)
   training_score = clf.score(X_train, y_train) test_score =
    clf.score(X_test, y_test) K.append(k)
    training.append(training_score)


    test.append(test_score)
    scores[k] = [training_score, test_score]
ax = sns.stripplot(training)
ax.set(xlabel='values of k', ylabel='Training Score')
plt.show()
ax = sns.stripplot(test)
ax.set(xlabel='values of k', ylabel='Test Score'
     plt.show()
       plt.scatter(K, training, color='k')
       plt.scatter(K, test, color='g')
       plt.show()
```

Output:

Viva Questions:

1. **Why do we need clustering in ML?**
2. **List the clustering model in machine learning**
3. **List the libraries for plotting graphs in ML**

Faculty Signature

Week11: Implementing K-means Clustering

```
# ----------installations---------------
# 1.pip install scikit-learn
# 2.pip install matplotlib
# 3.pip install k-means-constrained
#   4.pip   install   pandas
from       sklearn.datasets
import make_blobs
import matplotlib.pyplot as plt from
k_means_constrained import
KMeansConstrained import pandas as pd df =
pd.read_csv('student_clustering.csv') X =
df.iloc[:, :].values
km = KMeansConstrained(n_clusters=4, max_iter=500)
y_means = km.fit_predict(X)
plt.scatter(X[y_means == 0, 0], X[y_means == 0, 1], color='red')
plt.scatter(X[y_means == 1, 0], X[y_means == 1, 1], color='blue')
plt.scatter(X[y_means == 2, 0], X[y_means == 2, 1], color='green')
plt.scatter(X[y_means == 3, 0], X[y_means == 3, 1], color='yellow') plt.show()
```

Output:

Viva questions:

1. **What is k means clustering?**
2. **Give difference between k-means and KNN**
3. **How to import CSV files in python**

Faculty Signature

Week12: Implementing Hierarchical Clustering

To demonstrate the application of hierarchical clustering in Python, we will use the Iris dataset. The Iris dataset is one of the most common datasets that is used in machine learning for illustration purposes.
The Iris data has three types of Iris flowers which are three classes in the dependent variable. And it contains four independent variables which are sepal length, sepal width, petal length and petal width, all in cm. We will compare the original classes with the classes formed using hierarchical clustering methods.

Step 1 - Import data

We will import the dataset from the sklearn library.

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

# Import iris data
iris = datasets.load_iris()

iris_data = pd.DataFrame(iris.data)
iris_data.columns = iris.feature_names
iris_data['flower_type'] = iris.target
iris_data.head()
```

Output:

Step 2 - Visualise the classes

```
iris_X = iris_data.iloc[:, [0, 1, 2,3]].values
iris_Y = iris_data.iloc[:,4].values

import matplotlib.pyplot as plt
plt.figure(figsize=(10, 7))
plt.scatter(iris_X[iris_Y == 0, 0], iris_X[iris_Y == 0, 1], s=100, c='blue', label='Type 1')
plt.scatter(iris_X[iris_Y == 1, 0], iris_X[iris_Y == 1, 1], s=100, c='yellow', label='Type 2')
plt.scatter(iris_X[iris_Y == 2, 0], iris_X[iris_Y == 2, 1], s=100, c='green', label='Type 3')
plt.legend()
plt.xlabel('Sample Index')
plt.ylabel('Euclidean Distance')
plt.show()
```

Output:

Step 3 - Create a dendrogram

We start by importing the library that will help to create dendrograms. The dendrogram helps to give a rough idea of the number of clusters.

```
import scipy.cluster.hierarchy as sc
# Plot dendrogram
plt.figure(figsize=(20, 7))
plt.title("Dendrograms")

# Create dendrogram
sc.dendrogram(sc.linkage(iris_X, method='ward'))

plt.title('Dendrogram')
plt.xlabel('Sample index')
plt.ylabel('Euclidean distance')
```

Output:

Step 4 - Fit the model

We instantiate Agglomerative Clustering. Pass Euclidean distance as the measure of the distance between points and ward linkage to calculate clusters' proximity. Then we fit the model on our data points. Finally, we return an array of integers where the values correspond to the distinct categories using lables_ property.

```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(
    n_clusters=3, affinity='euclidean', linkage='ward')

cluster.fit(iris_X)
labels = cluster.labels_
labels
```

Output:

Step 5 - Visualise the cluster

```
plt.figure(figsize=(10, 7))
plt.scatter(iris_X[labels == 0, 0], iris_X[labels == 0, 1], s = 100, c = 'blue', label = 'Type 1')
plt.scatter(iris_X[labels == 1, 0], iris_X[labels == 1, 1], s = 100, c = 'yellow', label = 'Type 2')
plt.scatter(iris_X[labels == 2, 0], iris_X[labels == 2, 1], s = 100, c = 'green', label = 'Type 3')
plt.legend()
plt.xlabel('Sample Index')
plt.ylabel('Euclidean Distance')
plt.show()
```

Output:

Faculty Signature