**STELLA MARY’S COLLEGE OF ENGINEERING**

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai & Accredited by NAAC & NBA (CSE & MECH)

Aruthenganvilai, Kallukatti Junction, Azhikal Post Kanyakumari District – 629 202, Tamil Nadu



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CS3491 - Artificial Intelligence & Machine Learning Laboratory**

**REGULATION – 2021**

SEMESTER-IV

**2022-2023(Even)**



**LAB MANUAL**

Prepared by,

**Mrs. G. SANTHIYA, AP/ Dept. of CSE**

**Institute Vision:**

To emerge as a premiere institution, acknowledged as a center for excellence imparting technical education, creating technocrats who can address the needs of the society through exploration and experimentation and uplift mankind.

**Department Vision:**

To produce Computer Science professionals who can accomplish path-breaking solutions for a better society, through quality technical education, on gaining the required inter-personal, entrepreneurial and computing skills. .

**Institute Mission:**

To provide an education that transforms students, through rigorous course-work and by providing an understanding of the needs of the society and the industry.

**Department Mission:**

 To impart a holistic and experiential learning experience by making use of innovative teaching methodologies.

 To provide optimal technology solutions through collaborative and life-long learning for industry and societal needs with universal ethical values.

 To nurture leadership skills and facilitate various co-curricular and extra-curricular activities to implant the spirit of entrepreneurship.

 To provide industry-institute-interaction opportunities in order to motivate inter-disciplinary research capabilities with an inquiring mind.

Page2

**Programme Education Objectives (PEOs)**

PEO1:

Graduates will be competent in creating innovative technologies through inter-disciplinary research and comprehensive skills sets that are suitable for the global computing industry.

PEO2:

Graduates will be capable of managing leading positions with a broad understanding of

the application of ethics in evolving computer-based solutions for the societal needs.

PEO3:

Graduates will imbibe entrepreneurial qualities and develop their career by upgrading

their, communication, analytical and professional skills constantly.

**Programme Specific Outcomes(PSOs)**

PSO1:

Use data management techniques and algorithmic thinking for Software Design and

Development practices.

PSO2:

Develop reliable IT solutions based on the expertise in Distributed Applications Development, Web Designing and Networking for various societal needs and entrepreneurial practices ethically.

Page3

PSO3:

Manage multidisciplinary environments effectively through their interpersonal and

analytical skills and be responsible members and leaders of the society.

**PROGRAMME 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.

Page4

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.

Page5

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.

Page6

**RUBRICS FOR ASSESSING LABORATORY**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sl. | Criteria | Marks | Excellent (3) | Good (2) | Average (1) | Poor (0) |
| No. | 91% - 100% | 71% - 90% | 50% - 70% | <50% |
| 1 | Observation | 3 | Gives clear idea about the aim and having the capability of both recording & analyzing the data much easier. (3) | Capability of both recording & analyzing the data much easier but no proper clarification about the objective.  (2) | Gives clear idea about the target and has less capability of both recording & analyzing the data.  (1) | Gives indistinct idea about the target and has less capability of both recording & analyzing the data & who feel difficult to follow the objectives. (<1) |
| 2 | Assessment | 3 | Have executed the system in an efficient way & make credible and unbiased judgments regarding the conduct of the experiments.  (3) | Executed the system with less difficulty & has partial judgements regarding the overall system.  (2) | Executed the system with less efficiency and has no judgements regarding the system. (1) | Incomplete system execution & lack of judgments regarding the system. (<1) |
| 3 | Submission | 4 | Followed all the instructions given in the procedure and submitted the observation books in time. (4) | Followed all the instructions given in the procedure with some assisting (3) | Followed some of the instructions given in the procedure & late in submission of note books. (2) | Trying to follow the instructions given in the procedure & late in submission of note books. (<1) |

Page7

## COURSE OBJECTIVES

• The main objectives of this course are to:

• Study about uninformed and Heuristic search techniques.

• Learn techniques for reasoning under uncertainty

• Introduce Machine Learning and supervised learning algorithms

• Study about ensembling and unsupervised learning algorithms

• Learn the basics of deep learning using neural networks

**COURSEOUTCOMES**

CO1: Use appropriate search algorithms for problem solving

CO2: Apply reasoning under uncertainty

CO3: Build supervised learning models

CO4: Build ensembling and unsupervised models

CO5: Build deep learning neural network models

Page8

**PRACTICAL EXERCISES:**

1. Implementation of Uninformed search algorithms (BFS, DFS)

2. Implementation of Informed search algorithms (A\*, memory-bounded A\*)

3. Implement naïve Bayes models

4. Implement Bayesian Networks

5. Build Regression models

6. Build decision trees and random forests

7. Build SVM models

8. Implement ensembling techniques

9. Implement clustering algorithms

10. Implement EM for Bayesian networks

11. Build simple NN models

12. Build deep learning NN models

Page9

**CONTENTS**

**Name of the laboratory:** **Laboratory code:**

**Date** **Exp.**

**No.**

**Nameof theExperiment** **Page Marks**

**No.**

**Signatur e**

Page10

**CONTENTS**

**Nameof thelaboratory:** **Laboratory code:**

**Date** **Exp.**

**No.**

**Nameof theExperiment** **Page Marks**

**No.**

**Signatur e**

Page11

**Ex.No.1a:** **IMPLEMENTATION OF BASIC SEARCH STRATEGIES – BFS**

**AIM:**

To implement a python program for Breadth First Search (BFS)

**Breadth-First Search**

Breadth-first search (BFS) is a traversing algorithm which starts from a selected node (source or starting node) and explores all of the neighbour nodes at the present depth before moving on to the nodes at the next level of depth.

 It must be ensured that each vertex of the graph is visited exactly once to avoid getting into an infinite loop with cyclic graphs or to prevent visiting a given node multiple times when it can be reached through more than one path.

Breadth-first search can be implemented using a [queue data structure,](https://open4tech.com/stacks-vs-queues/) which follows the first-in-first-out (FIFO) method – i.e., the node that was inserted first will be visited first, and so on.

**ALGORITHM**:

Step 1: We start the process by considering any random node as the starting vertex.

Step 2: We enqueue (insert) it to the queue and mark it as visited.

Step 3: Then we mark and enqueue all of its unvisited neighbours at the current depth or continue to the next depth level if there is any.

Step 4: The visited vertices are removed from the queue. Step 5:The process ends when the queue becomes empty.

Page12

**PROGRAM**

graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[]

}

visited =[]

queue=[]

def bfs(visited,graph,node):

visited.append(node)

queue.append(node)

while queue:

m=queue.pop(0)

print(m, end="")

for neighbour in graph[m]:

if neighbour not in visited:

visited.append(neighbour)

queue.append(neighbour)

print ("Following is the Breadth-First Search")

bfs(visited,graph,'5')

Page13

**OUTPUT:**

Following is the Breadth-First Search 537248

**RESULT:**

Thus the program for breadth-first search was implemented and executed successfully.

Page14

**Ex.No.1b:** **IMPLEMENTATION OF BASIC SEARCH STRATEGIES – DFS**

**AIM:**

To implement a python code for Depth First Search (DFS)

**ALGORITHM:**

Step: 1 Pick any node. If it is unvisited, mark it as visited and recur on all its adjacent nodes.

Step: 2 Repeat until all the nodes are visited, or the node to be searched is found.

visited is a set that is used to keep track of visited nodes.

The dfs function is called and is passed the visited set, the graph in the form of a dictionary, and A, which is the starting node.

dfs follows the algorithm described above:

1. It first checks if the current node is unvisited - if yes, it is appended in the visited set.

2. Then for each neighbor of the current node, the dfs function is invoked again. 3. The base case is invoked when all the nodes are visited. The function then

returns.

**PROGRAM** graph = {

'5' : ['3','7'], '3' : ['2', '4'], '7' : ['8'],

'2' : [], '4' : ['8'], '8' : []

}

visited = set()

def dfs(visited, graph, node): #function for dfs if node not in visited:

print (node) visited.add(node)

for neighbour in graph[node]: dfs(visited, graph, neighbour)

Page15

print("Following is the Depth-First Search") dfs(visited, graph, '5')

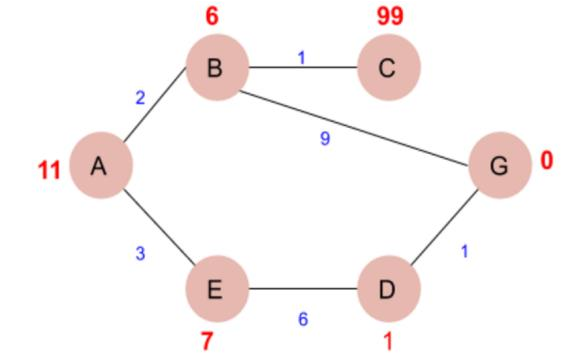
**OUTPUT:**

Following is the Depth-First Search 5

3 2 4 8 7

**RESULT:**

Thus the program for depth-first search was implemented and executed successfully. Page16



**Ex.No:2a** **IMPLEMENTATION OF A\* SEARCH ALGORITHM**

**AIM:**

To implement a path finding using A\* search algorithm.

**A\* SEARCH :**

A\* search finds the shortest path through a search space to the goal state using the heuristic function.

This technique finds minimal cost solutions and is directed to a goal state called A\* search.

The A\* algorithm also finds the lowest-cost path between the start and goal state, where changing from one state to another requires some cost.

**STEPS FOR SOLVING A\* SEARCH**

Given the graph, find the cost-effective path from A to G. That is A is the source node and G is the goal node.

Now from A, we can go to point B or E, so we compute f(x) for each of them,

A → B = g(B) + h(B) = 2 + 6 = 8

A → E = g(E) + h(E) = 3 + 7 = 10

Since the cost for A → B is less, we move forward with this path and compute the f(x) for the children nodes of B.

Now from B, we can go to point C or G, so we compute f(x) for each of them,

A → B → C = (2 + 1) + 99= 102

A → B → G = (2 + 9 ) + 0 = 11

Here the path A → B → G has the least cost but it is still more than the cost of A → E, thus we explore this path further.

Page17

Now from E, we can go to point D, so we compute f(x),

A → E → D = (3 + 6) + 1 = 10

Comparing the cost of A → E → D with all the paths we got so far and as this cost is least of all we move forward with this path.

Now compute the f(x) for the children of D

A → E → D → G = (3 + 6 + 1) +0 = 10

Now comparing all the paths that lead us to the goal, we conclude that **A → E → D → G** is the most cost-effective path to get from A to G.

**ALGORITHM:**

// A\* Search Algorithm

Step 1: Place the starting node into OPEN and find its f (n) value.

Step 2: Remove the node from OPEN, having the smallest f (n) value. If it is a goal node then stop and return success.

Step 3: Else remove the node from OPEN, find all its successors.

Step 4: Find the f (n) value of all successors; place them into OPEN and place the removed node into CLOSE.

Step 5: Go to Step-2.

Step 6: Exit.

**PROGRAM:**

def aStarAlgo(start\_node, stop\_node):

open\_set = set(start\_node)

closed\_set = set()

g = {}

parents = {}

#store distance from starting node

# parents contains an adjacency map of all nodes

#distance of starting node from itself is zero

g[start\_node] = 0

#start\_node is root node i.e it has no parent nodes Page18 #so start\_node is set to its own parent node

parents[start\_node] = start\_node

while len(open\_set) > 0:

n = None

#node with lowest f() is found

for v in open\_set:

if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

n = v

if n == stop\_node or Graph\_nodes[n] == None:

pass

else:

for (m, weight) in get\_neighbors(n):

#nodes 'm' not in first and last set are added to first

#n is set its parent

if m not in open\_set and m not in closed\_set:

open\_set.add(m)

parents[m] = n

g[m] = g[n] + weight

#for each node m,compare its distance from start i.e g(m) to the

#from start through n node

else:

if g[m] > g[n] + weight:

#update g(m)

g[m] = g[n] + weight

#change parent of m to n

parents[m] = n

#if m in closed set,remove and add to open

if m in closed\_set:

closed\_set.remove(m)

open\_set.add(m)

if n == None: Page19

print('Path does not exist!')

return None

# if the current node is the stop\_node

# then we begin reconstructin the path from it to the start\_node

if n == stop\_node:

path = []

while parents[n] != n:

path.append(n)

n = parents[n]

path.append(start\_node)

path.reverse()

print('Path found: {}'.format(path))

return path

# remove n from the open\_list, and add it to closed\_list

# because all of his neighbors were inspected

open\_set.remove(n)

closed\_set.add(n)

print('Path does not exist!')

return None

#define fuction to return neighbor and its distance

#from the passed node

def get\_neighbors(v):

if v in Graph\_nodes:

return Graph\_nodes[v]

else:

return None

#for simplicity we ll consider heuristic distances given

#and this function returns heuristic distance for all nodes Page20

def heuristic(n):

H\_dist = {

'A': 11,

'B': 6,

'C': 5,

'D': 7,

'E': 3,

'F': 6,

'G': 5,

'H': 3,

'I': 1,

'J': 0

}

return H\_dist[n]

#Describe your graph here

Graph\_nodes = {

'A': [('B', 6), ('F', 3)],

'B': [('A', 6), ('C', 3), ('D', 2)],

'C': [('B', 3), ('D', 1), ('E', 5)],

'D': [('B', 2), ('C', 1), ('E', 8)],

'E': [('C', 5), ('D', 8), ('I', 5), ('J', 5)],

'F': [('A', 3), ('G', 1), ('H', 7)],

'G': [('F', 1), ('I', 3)],

'H': [('F', 7), ('I', 2)],

'I': [('E', 5), ('G', 3), ('H', 2), ('J', 3)],

}

aStarAlgo('A', 'J')

Page21

**OUTPUT:**

Path found: ['A', 'F', 'G', 'I', 'J']

**RESULT:**

Thus the program for A\* search algorithm for path was implemented and executed successfully.

Page22

**Ex.No:2b** **IMPLEMENTATION OF MEMORY BOUNDED A\* ALGORITHM**

**AIM:**

To implement memory bounded A\* search for path finding problem.

**Memory bounded A\* Search:**

Memory Bounded A\* is **a shortest path algorithm based on the A\* algorithm**. The main advantage is that it uses a bounded memory, while the A\* algorithm might

need exponential memory. All other characteristics of are inherited from A\*.

This search is an optimal and complete algorithm for finding a least-cost path. Unlike A\*, it will not run out of memory, unless the size of the shortest path exceeds the amount of space in available memory.

**STEPS FOR MEMORY BOUND SEARCH**

Step 1: Works like A\* until memory is full

Step 2: When memory is full, drop the leaf node with the highest f-value (the worst leaf), keeping track of that worst value in the parent

Step 3: Complete if any solution is reachable

Step 4: Optimal if any optimal solution is reachable

Step 5: Otherwise, returns the best reachable solution

**ALGORITHM:**

function SMA\*(problem)return a solution sequence

input: problem, a problem

static: Queue, a queue of nodes ordered by f-cost

QueueMake –Queue ({Make-Node(INITIAL-STATE(problem)})

Loop do

If queue = empty return false

Ndeepest least-f cost node in queue

If GOAL-TEST(n) return true

SNEXT-SUCCESSOR(n)

if s≠ goal and is maximum depth

f(s)*∞*

else

f(s)Max (f(n),g(n)+h(n)) Page23

if all of n’s successor have been generated

update n’s f-cost and those of its ancestors if necessary

if successors(n) all in memory then remove n from Queue

if memory = full

delete shallowest,highest-f-cost node in Queue

remove it from its parenr’s successor list

insert its parent on Queue if necessary

insert s on Queue

end

**PROGRAM:**

class Graph:

def \_\_init\_\_(self, adjac\_lis):

self.adjac\_lis = adjac\_lis

self.H1 = {

'A': 1,

'B': 6,

'C': 2,

'D': 2,

'E': 2,

'F': 1,

'G': 5,

'H': 7,

'I': 7,

'J': 1,

'T': 3

}

self.H = { Page24

'A': 1,

'B': 6,

'C': 12,

'D': 10,

'E': 4,

'F': 4,

'G': 5,

'H': 7,

}

self.parent={}

self.openList=set()

self.hasRevised=[]

self.solutionGraph={}

self.solvedNodeList=set()

def get\_neighbors(self, v):

return self.adjac\_lis.get(v,'')

def updateNode(self, v):

if v in self.solvedNodeList:

return

feasibleChildNodeList=[]

minimumCost=None

minimumCostFeasibleChildNodesDict={} Page25

print("CURRENT PROCESSING NODE:", v)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#computing the minimum cost by visiting all child nodes with "OR/AND" condition

for (c, weight) in self.get\_neighbors(v):

feasibleChildNodeList=[] #initialize for computing any new feasibileChildNodes(childnodes with minimum cost)

cost= self.getHeuristicNodeValue(c) + 1 # assuming all the edges with equal weight one

feasibleChildNodeList.append(c)

andNodesList=self.getAndNodes(v)

for nodeTuple in andNodesList: #checking whether the child(c) is in "AND" condition with other nodes

if c in nodeTuple:

for andNode in nodeTuple:

if andNode!=c:

feasibleChildNodeList.append(andNode)

cost=cost+self.getHeuristicNodeValue(andNode) + 1 #compute total cost of "AND" nodes

if minimumCost==None: #inializing minimum cost

minimumCost=cost

for child in feasibleChildNodeList: #capturing parent child relationship

self.parent[child]=v

minimumCostFeasibleChildNodesDict[minimumCost]=feasibleChildNodeList

#mapping minimum cost child nodes Page26

else:

if minimumCost>cost: #checking minimum cost child nodes

minimumCost=cost

for child in feasibleChildNodeList:

self.parent[child]=v

minimumCostFeasibleChildNodesDict[minimumCost]=feasibleChildNodeList

if minimumCost==None: # no child nodes of the give node v and mark as solved node

minimumCost=self.getHeuristicNodeValue(v)

self.solvedNodeList.add(v)

else:

self.setHeuristicNodeValue(v,minimumCost) # minimum cost found! assign to the given node v

for child in minimumCostFeasibleChildNodesDict[minimumCost]:

if child not in self.solvedNodeList: nodes are solved

self.openList.add(child) exploration

# checking whether minimum cost child

# if not solved add node/s to openList for

self.solutionGraph[v]= minimumCostFeasibleChildNodesDict[minimumCost]

# capture minimum cost child nodes as part of graph solution

solved=True

for c in self.solutionGraph.get(v,''): nodes are already solved

if c not in self.solvedNodeList:

solved=solved & False

# checking if feasible minimum cost child

Page27

if solved == True: are solved

self.solvedNodeList.add(v)

# if all the feasible child nodes of the given node

# mark the given node v as solved

print("HEURISTIC VALUES :", self.H)

print("OPEN LIST :", list(self.openList))

print("MINIMUM COST NODES:", minimumCostFeasibleChildNodesDict.get(minimumCost,"[ ]"))

print("SOLVED NODE LIST :", list(self.solvedNodeList))

print("-----------------------------------------------------------------------------------------")

def getAndNodes(self,v):

andNodes={

# PARENT NODE AS KEY FOR ITS CHILD NODES IN AND CONDITION

'A':[('B','C')],

'D':[('E','F')]

}

return andNodes.get(v, '')

def getHeuristicNodeValue(self, n):

return self.H.get(n,0) #Always return the heuristic value

def setHeuristicNodeValue(self, n, value):

self.H[n]=value # sets the revised heuristic value of a give node

def ao\_star\_algorithm(self, start):

# In this open\_lst is a list of nodes which have been visited, but who's

# neighbours haven't all been always inspected, It starts off with the start node. Page28

# And closedList is a list of nodes which have been visited and who's neighbors have been always inspected.

self.openList = set([start])

while len(self.openList) > 0:

# it will find a node with the lowest value of f() -

v = self.openList.pop() # procesess a Node in the openList

self.updateNode(v)

while v!=start and self.parent[v] not in self.solvedNodeList:

parent=self.parent[v];

self.updateNode(parent)

v=parent

print("TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH")

print("---------------------------------------------------------------")

print("SOLUTION GRAPH:",self.solutionGraph)

print("\n")

nodeList1 = {

'A': [('B', 1), ('C', 1), ('D', 1)],

'B': [('G', 1), ('H', 1)],

'C': [('J', 1)],

'D': [('E', 1), ('F', 1)],

'G': [('I', 1)]

Page29 }

nodeList = {

'A': [('B', 1), ('C', 1), ('D', 1)],

'B': [('G', 1), ('H', 1)],

'D': [('E', 1), ('F', 1)]

}

graph = Graph(nodeList)

graph.ao\_star\_algorithm('A')

**OUTPUT:**

CURRENT PROCESSING NODE: A

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['D']

MINIMUM COST NODES: ['D']

SOLVED NODE LIST : []

-----------------------------------------------------------------------------------------

TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH

---------------------------------------------------------------

SOLUTION GRAPH: {'A': ['D']}

CURRENT PROCESSING NODE: D

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['F', 'E']

MINIMUM COST NODES: ['E', 'F']

SOLVED NODE LIST : []

-----------------------------------------------------------------------------------------

CURRENT PROCESSING NODE: A Page30

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['F', 'E', 'D']

MINIMUM COST NODES: ['D']

SOLVED NODE LIST : []

-----------------------------------------------------------------------------------------

TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH

---------------------------------------------------------------

SOLUTION GRAPH: {'A': ['D'], 'D': ['E', 'F']}

CURRENT PROCESSING NODE: F

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['E', 'D']

MINIMUM COST NODES: [ ]

SOLVED NODE LIST : ['F']

-----------------------------------------------------------------------------------------

CURRENT PROCESSING NODE: D

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['E', 'D']

MINIMUM COST NODES: ['E', 'F']

SOLVED NODE LIST : ['F']

-----------------------------------------------------------------------------------------

CURRENT PROCESSING NODE: A

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['E', 'D']

MINIMUM COST NODES: ['D']

SOLVED NODE LIST : ['F']

----------------------------------------------------------------------------------------- Page31

TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH

---------------------------------------------------------------

SOLUTION GRAPH: {'A': ['D'], 'D': ['E', 'F']

CURRENT PROCESSING NODE: E

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['D']

MINIMUM COST NODES: [ ]

SOLVED NODE LIST : ['F', 'E']

-----------------------------------------------------------------------------------------

CURRENT PROCESSING NODE: D

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['D']

MINIMUM COST NODES: ['E', 'F']

SOLVED NODE LIST : ['F', 'D', 'E']

-----------------------------------------------------------------------------------------

CURRENT PROCESSING NODE: A

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEURISTIC VALUES : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}

OPEN LIST : ['D']

MINIMUM COST NODES: ['D']

SOLVED NODE LIST : ['F', 'D', 'A', 'E']

-----------------------------------------------------------------------------------------

TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH

---------------------------------------------------------------

SOLUTION GRAPH: {'A': ['D'], 'D': ['E', 'F']}

TRAVERSE SOLUTION FROM ROOT TO COMPUTE THE FINAL SOLUTION GRAPH

---------------------------------------------------------------

SOLUTION GRAPH: {'A': ['D'], 'D': ['E', 'F']} Page32

**RESULT:**

Thus the program for memory bounded A\* search was implemented and executed successfully.

Page33

**Ex.No:3** **IMPLEMENT NAÏVE BAYES MODEL**

**AIM:**

To implement a program for Naïve Bayes model

**NAÏVE BAYES CLASSIFIER ALGORITHM**

Naive Bayes is among one of the very simple and powerful algorithms for classification based on Bayes Theorem with an assumption of independence among the predictors.

 The Naive Bayes classifier assumes that the presence of a feature in a class is not related to any other feature.

Naive Bayes is a classification algorithm for binary and multi-class classification problems.

**Bayes Theorem**

 Based on prior knowledge of conditions that may be related to an event, Bayes theorem describes the probability of the event

 conditional probability can be found this way

 Assume we have a Hypothesis(*H*) and evidence(*E*),

 According to Bayes theorem, the relationship between the probability of Hypothesis before getting the evidence represented as *P(H)* and the probability of the hypothesis after getting the evidence represented

as P(H|E) is:

P(H|E) = P(E|H)\*P(H)/P(E)

**STEPS INVOLVE NAÏVE BAYES ALGORITHM**

**Step 1: Handling Data**

Data is loaded from the .csv file and spread into training and tested assets.

**Step 2: Summarizing the data**

Summarise the properties in the training data set to calculate the probabilities and make predictions.

**Step 3: Making a Prediction**

A particular prediction is made using a summarise of the data set to make a single prediction

**Step 4: Making all the Predictions**

Generate prediction given a test data set and a summarise data set.

**Step 4: Evaluate Accuracy:**

Accuracy of the prediction model for the test data set as a percentage correct out of them all

the predictions made. Page34

**Step 4: Trying all together**

Finally, we tie to all steps together and form our own model of Naive Bayes Classifier.

**PROGRAM:**

import pandas as pd

msg=pd.read\_csv('C:/python/naivetext.csv',names=['message','label'])

print('The dimensions of the dataset',msg.shape)

msg['labelnum']=msg.label.map({'pos':1,'neg':0})

X=msg.message

y=msg.labelnum

print(X)

print(y)

**OUTPUT:**

The dimensions of the dataset (18, 2) 0 I love this sandwich

1 This is an amazing place

2 I feel very good about these beers 3 This is my best work

4 What an awesome view 5 I do not like this restaurant

6 I am tired of this stuff 7 I can't deal with this

8 He is my sworn enemy 9 My boss is horrible

10 This is an awesome place 11 I do not like the taste of this juice 12 I love to dance

13 I am sick and tired of this place 14 What a great holiday 15 That is a bad locality to stay

16 We will have good fun tomorrow 17 I went to my enemy's house today Name: message, dtype: object

0 1 1 1 2 1 3 1 4 1 5 0

6 0 Page35 7 0

8 0

9 0 10 1 11 0 12 1 13 0 14 1 15 0 16 1 17 0

Name: labelnum, dtype: int64

from sklearn.model\_selection import train\_test\_split

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

print ('\n The total number of Training Data :',ytrain.shape)

print ('\n The total number of Test Data :',ytest.shape)

**OUTPUT:**

The total number of Training Data : (13,)

The total number of Test Data : (5,)

from sklearn.feature\_extraction.text import CountVectorizer

count\_vect = CountVectorizer()

xtrain\_dtm = count\_vect.fit\_transform(xtrain)

xtest\_dtm=count\_vect.transform(xtest)

print('\n The words or Tokens in the text documents \n')

print(count\_vect.get\_feature\_names())

df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names())

from sklearn.naive\_bayes import MultinomialNB

clf = MultinomialNB().fit(xtrain\_dtm,ytrain)

predicted = clf.predict(xtest\_dtm)

Page36

from sklearn import metrics

print('\n Accuracy of the classifer is',

metrics.accuracy\_score(ytest,predicted))

**OUTPUT:**

Accuracy of the classifer is 0.8

print('\n Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

print('\n The value of Precision' ,

metrics.precision\_score(ytest,predicted))

print('\n The value of Recall' ,

metrics.recall\_score(ytest,predicted))

**OUTPUT:** Confusion matrix [[3 0]

[1 1]]

The value of Precision 1.0

The value of Recall 0.5

**RESULT:**

Thus the program for naïve Bayes model was implemented and executed successfully.

Page37

**Ex.No:4** **Implement Bayesian networks**

**AIM:**

To write a program to construct a Bayesian network to diagnose CORONA infection

using standard WHO Data set.

**Data set:**

Attribute Information:

1. Breathing Problem 2. Fever

3. Dry cough 4. Sore throat

5. Running Nose 6. Asthma

7. Chronic Lung disease 8. Headache

9. Heart disease 10. Diabetes

11. Hyper tension

**PROGRAM:**

import pandas as pd

import numpy as np

import time

import json

def pre\_processing(df):

print("Processing Test Data...")

time.sleep(2)

x = df.drop(df.columns[-1],axis=1)

y = df[df.columns[-1]]

print("Done!\n") Page38

return x, y

def test\_model(df,x,y,testData,trained\_data):

print(f"Model Testing In Progress...")

time.sleep(3)

total=len(df[y.name])

totalYes=trained\_data["totalYes"]

totalNo=trained\_data["totalNo"]

sum1Yes=(totalYes/total)

sum2Yes=1

for indx2 in range(len(testData)):

sum1Yes\*=(trained\_data[x.columns[indx2]][testData[indx2]]['yes']/totalYes)

sum2Yes\*=(trained\_data[x.columns[indx2]][testData[indx2]]['total']/total)

sumYes = sum1Yes/sum2Yes

sum1No=(totalNo/total)

sum2No=1

for indx2 in range(len(testData)):

sum1No\*=(trained\_data[x.columns[indx2]][testData[indx2]]['no']/totalNo)

sum2No\*=(trained\_data[x.columns[indx2]][testData[indx2]]['total']/total)

sumNo = sum1No/sum2No

print(f"\nProbability of Covid 19 to be True : {format(sumYes,'.2f')}")

print(f"Probability of Covid 19 to be False : {format(sumNo,'.2f')}")

if sumYes>sumNo:

ans="Yes"

else:

ans="No"

print(f"Answer is - {ans}\n\n\n")

def read\_data():

print("Reading data from csv...")

time.sleep(2) Page39 df = pd.read\_csv("C:/python/covid.csv")

print("Done...\n")

return df

def load\_model():

print("Loading Trained Data from trained\_data.txt...")

time.sleep(2)

with open('C:/python/trained\_data.txt') as f:

data = f.read()

js = json.loads(data)

print("Loaded!\n")

return js

df = pd.DataFrame(read\_data())

x,y=pre\_processing(df)

x=x.truncate(0,4433)

y=y.truncate(0,4433)

trained\_data=load\_model()

while True:

print("\nEnter you choice of features:\n")

test=[]

for uniqueCol in list(x.columns):

columnData=list(np.unique(df[uniqueCol]))

for idx in range(len(columnData)):

print(f"{idx}. {columnData[idx]}")

inp=int(input(f"Enter option for {uniqueCol}: "))

test.append(columnData[inp])

print("\n")

print(f"Your test data is: {test}\n")

test\_model(df,x,y,test,trained\_data)

Page40

**OUTPUT:**

Reading data from csv... Done...

Processing Test Data... Done!

Loading Trained Data from trained\_data.txt... Loaded!

Enter you choice of features:

0. No 1. Yes

Enter option for Breathing Problem: 1

0. No 1. Yes

Enter option for Fever: 1

0. No 1. Yes

Enter option for Dry Cough: 1

0. No 1. Yes

Enter option for Sore throat: 1

0. No 1. Yes

Enter option for Running Nose: 0

0. No 1. Yes

Enter option for Asthma: 0

0. No 1. Yes

Enter option for Chronic Lung Disease: 0

Page41 0. No

1. Yes

Enter option for Headache: 1

0. No 1. Yes

Enter option for Heart Disease: 0

0. No 1. Yes

Enter option for Diabetes: 0

0. No 1. Yes

Enter option for Hyper Tension: 0

0. No 1. Yes

Enter option for Abroad travel: 1

0. No 1. Yes

Enter option for Contact with COVID Patient: 0

0. No 1. Yes

Enter option for Attended Large Gathering: 1

0. No 1. Yes

Enter option for Visited Public Exposed Places: 0

0. No 1. Yes

Enter option for Family working in Public Exposed Places: 1

0. No 1. Yes

Enter option for Wearing Masks: 1

Page42 0. No

1. Yes

Enter option for Sanitization from Market: 1

Your test data is: ['Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Y es', 'No', 'Yes', 'Yes', 'Yes']

Model Testing In Progress...

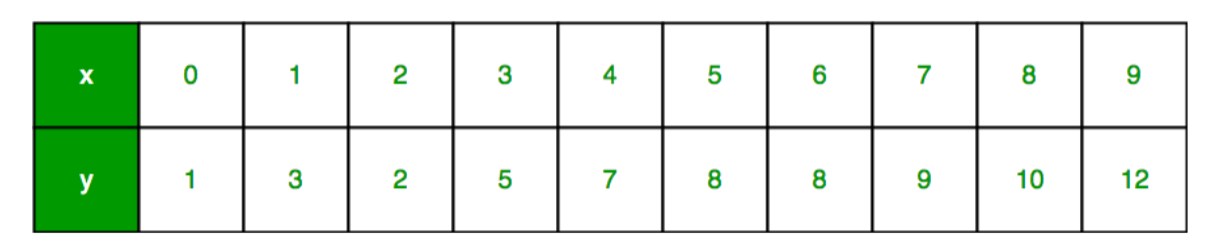
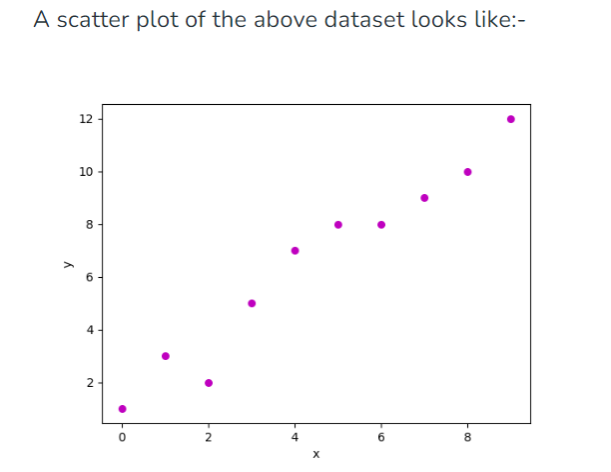
Probability of Covid 19 to be True : 0.92 Probability of Covid 19 to be False : 0.00 Answer is - Yes

**RESULT:**

Thus the program to construct a Bayesian network was implemented and executed

successfully.

Page43



**Ex.No:5a Implement Regression models (Linear Regression)**

**AIM:**

To write a program to implement linear for modeling relationships between a

dependent variable with a given set of independent variables.

**DEFINITION:**

Let us consider a dataset where we have a value of response y for every feature x:

Now, the task is to find a **line that fits best** in the above scatter plot so that we can predict the response for any new feature values. (i.e a value of x not present in a dataset)

This line is called a **regression line**.

Page44

**PROGRAM**

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():

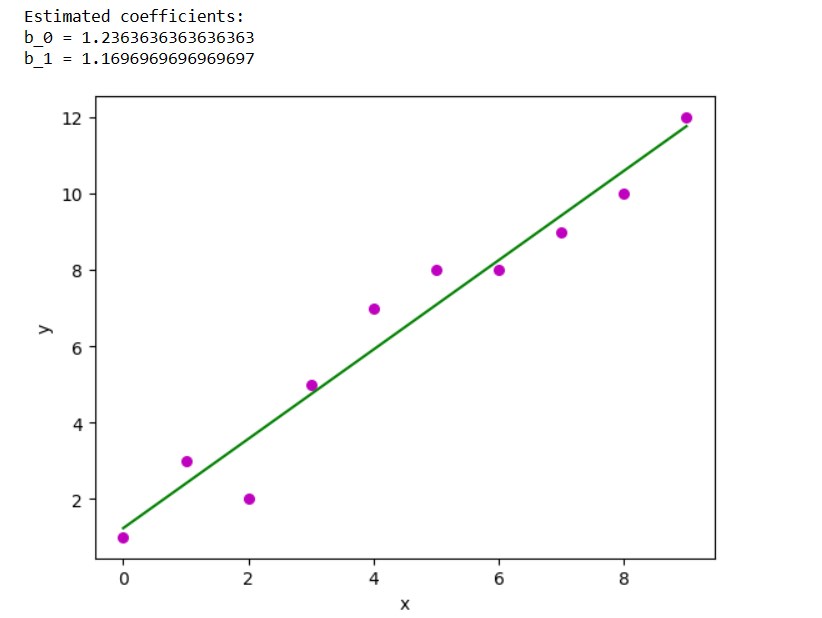
# observations / data

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 = {} \ Page45 \nb\_1 = {}".format(b[0], b[1]))



# plotting regression line plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_": main()

**OUTPUT:**

**RESULT:**

Thus the program to implement linear regression model was implemented and

executed successfully.

Page46

**Ex.No:5b Implement Regression models (Logistic Regression)**

**AIM:**

To implement Logistic Regression using Python

**ALGORITHM**

Import all the library function

Import make\_classification from sklearn datasets Generate Dataset for Logistic Regression Import pyplot from matplotlib

Classify the Dataset based on the given features.

**PROGRAM:**

from sklearn.datasets import make\_classification

from matplotlib import pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix

import pandas as pd

# Generate and dataset for Logistic Regression

x, y = make\_classification(

n\_samples=100,

n\_features=1,

n\_classes=2,

n\_clusters\_per\_class=1,

flip\_y=0.03,

n\_informative=1,

n\_redundant=0,

n\_repeated=0

)

print(x,y)

**OUTPUT:**

[[ 0.68072366] Page47 [-0.806672 ]

[-0.25986635] [-0.96951576] [-1.55870949] [-0.71107565] [ 0.05858082] [-2.06472972] [-0.61592043] [ 1.25423915] [ 0.81852686] [-1.65141186] [-0.5894455 ] [ 1.02745431] [-0.32508896] [-0.53886171] [ 1.14821234] [ 0.87538478] [ 0.95887802] [ 1.30514551] [-1.02478688] [ 0.16563384] [ 0.77626036] [-1.00622251] [-0.55976575] [ 1.33550038] [ 1.60327317] [ 1.82115858] [-0.68603388] [ 1.8733355 ] [-0.52494619] [-2.03314002] [ 0.47001797] [ 1.55400671] [-1.34062378] [-0.38624537] [-1.06339387] [-1.41465045] [ 0.58850401] [ 0.80925135] [-0.82066568] [-0.01262654] [-0.75104194] [-1.09609801] [-0.30652093] [-0.6945338 ] [-0.90156651] [-0.96587756] [ 0.53851931] [ 0.16533166] [-1.04609567] [-1.15065139]

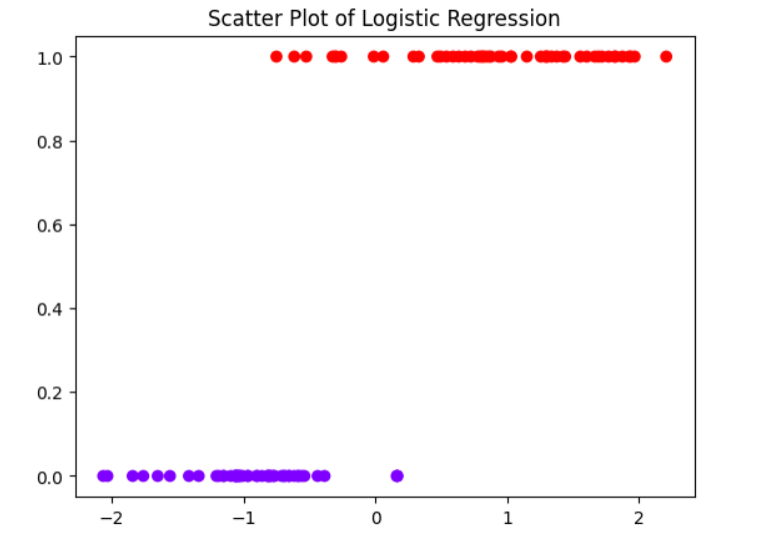
Page48

[-0.76739642] [ 0.83776929] [ 2.20562241] [-0.80368921] [-0.86160904] [ 0.86032131] [-0.65752318] [ 1.81228279] [-0.81507664] [ 0.93532773] [ 1.76874632] [ 0.32893072] [ 1.02960085] [-1.84150254] [ 0.16156709] [-1.05944665] [ 0.28788136] [-1.05549933] [ 1.37528673] [ 1.66369265] [ 1.71761177] [ 1.96597594] [-0.65315492] [-0.29598263] [-1.15345006] [-1.03851861] [ 1.69109822] [ 1.92402678] [-0.89593983] [-0.58208549] [-1.18750595] [-1.06231671] [-0.79230653] [ 1.42147278] [ 1.2887393 ] [ 1.93706073] [-1.03110736] [-1.20543711] [ 0.79446549] [ 1.29599432] [ 0.49396915] [ 0.63241066] [ 0.72416825] [-1.76099355] [-0.61639759] [-0.43854548] [ 1.43886371]

[-0.77167438]] [1 0 1 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1

0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0]

Page49



# Create a scatter plot

plt.scatter(x, y, c=y, cmap='rainbow') plt.title('Scatter Plot of Logistic Regression') plt.show()

# Split the dataset into training and test dataset

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=1)

x\_train.shape

**OUTPUT:**

(75, 1)

log\_reg=LogisticRegression()

log\_reg.fit(x\_train, y\_train)

y\_pred=log\_reg.predict(x\_test)

confusion\_matrix(y\_test, y\_pred)

Page50

**OUTPUT:**

array([[12, 0],

[ 2, 11]], dtype=int64)

**RESULT:**

Thus the program to implement linear regression model was implemented and

executed successfully.

Page51

**Ex.No:6a Implement Decision Tree**

**AIM:**

**ALGORITHM**

Import Decision tree classifier from sklearn model

Import train\_test\_split from sklearn.model Import accuracy\_score from sklearn.metrics Import classification\_report from sklearn.metrics Read the dataset values from the provided URL Print the dataset shape

Print the dataset observation Separate the target variable Splitting the dataset into train and test Perform training with giniIndex Creating the classifier object Perform training with entropy

Create Function to make prediction from the given dataset Create Function to calculate accuracy from the given dataset.

**PROGRAM**

import numpy as np

import pandas as pd

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

def importdata():

balance\_data = pd.read\_csv(

'https://archive.ics.uci.edu/ml/machine-learning-'+

'databases/balance-scale/balance-scale.data', Page52

sep= ',', header = None)

# Printing the dataswet shape

print ("Dataset Length: ", len(balance\_data))

print ("Dataset Shape: ", balance\_data.shape)

# Printing the dataset obseravtions

print ("Dataset: ",balance\_data.head())

return balance\_data

def splitdataset(balance\_data):

# Separating the target variable

X = balance\_data.values[:, 1:5]

Y = balance\_data.values[:, 0]

# Splitting the dataset into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, Y, test\_size = 0.3, random\_state = 100)

return X, Y, X\_train, X\_test, y\_train, y\_test

# Function to perform training with giniIndex.

def train\_using\_gini(X\_train, X\_test, y\_train):

# Creating the classifier object

clf\_gini = DecisionTreeClassifier(criterion = "gini",

random\_state = 100,max\_depth=3, min\_samples\_leaf=5)

# Performing training

clf\_gini.fit(X\_train, y\_train) Page53

return clf\_gini

# Function to perform training with entropy.

def tarin\_using\_entropy(X\_train, X\_test, y\_train):

# Decision tree with entropy

clf\_entropy = DecisionTreeClassifier(

criterion = "entropy", random\_state = 100,

max\_depth = 3, min\_samples\_leaf = 5)

# Performing training

clf\_entropy.fit(X\_train, y\_train)

return clf\_entropy

# Function to make predictions

def prediction(X\_test, clf\_object):

# Predicton on test with giniIndex

y\_pred = clf\_object.predict(X\_test)

print("Predicted values:")

print(y\_pred)

return y\_pred

# Function to calculate accuracy

def cal\_accuracy(y\_test, y\_pred):

print("Confusion Matrix: ",

confusion\_matrix(y\_test, y\_pred))

Page54

print ("Accuracy : ",

accuracy\_score(y\_test,y\_pred)\*100)

print("Report : ",

classification\_report(y\_test, y\_pred))

# Driver code

def main():

# Building Phase

data = importdata()

X, Y, X\_train, X\_test, y\_train, y\_test = splitdataset(data)

clf\_gini = train\_using\_gini(X\_train, X\_test, y\_train)

clf\_entropy = tarin\_using\_entropy(X\_train, X\_test, y\_train)

# Operational Phase

print("Results Using Gini Index:")

# Prediction using gini

y\_pred\_gini = prediction(X\_test, clf\_gini)

cal\_accuracy(y\_test, y\_pred\_gini)

print("Results Using Entropy:")

# Prediction using entropy

y\_pred\_entropy = prediction(X\_test, clf\_entropy)

cal\_accuracy(y\_test, y\_pred\_entropy)

# Calling main function

if \_\_name\_\_=="\_\_main\_\_": Page55

main()

**OUTPUT:**

Dataset Length: 625 Dataset Shape: (625, 5) Dataset: 0 1 2 3 4 0 B 1 1 1 1

1 R 1 1 1 2 2 R 1 1 1 3 3 R 1 1 1 4 4 R 1 1 1 5

Results Using Gini Index: Predicted values:

['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']

Confusion Matrix: [[ 0 6 7] [ 0 67 18]

[ 0 19 71]]

Accuracy : 73.40425531914893

Report : precision recall f1-score support

B 0.00 0.00 0.00 13 L 0.73 0.79 0.76 85 R 0.74 0.79 0.76 90

accuracy macro avg

weighted avg

0.73 188 0.49 0.53 0.51 188

0.68 0.73 0.71 188

Results Using Entropy: Predicted values:

['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'

Page56

'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']

Confusion Matrix: [[ 0 6 7] [ 0 63 22]

[ 0 20 70]]

Accuracy : 70.74468085106383

Report : precision recall f1-score support

B 0.00 0.00 0.00 13 L 0.71 0.74 0.72 85 R 0.71 0.78 0.74 90

accuracy macro avg

weighted avg

0.71 188 0.47 0.51 0.49 188

0.66 0.71 0.68 188

**RESULT:**

Thus the program to implement decision tree was implemented and executed

successfully. Page57



**Ex.No:6b Implement Random Forest**

**AIM:**

Import Load digits from sklearn.datasets

Import Random forest classifier from sklearn datasets Train the given dataset using Random Forest Classifier. Obtain the score from the trained dataset

**PROGRAM**

import pandas as pd

from sklearn.datasets import load\_digits

digits = load\_digits()

dir(digits)

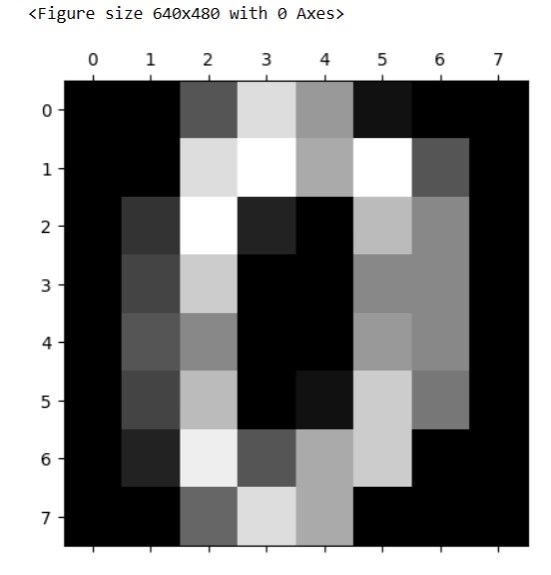
%matplotlib inline

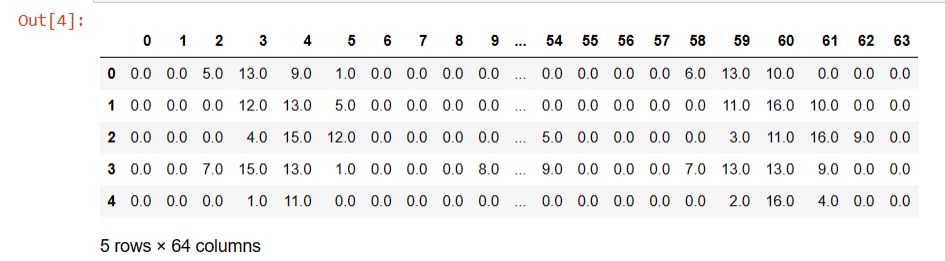
import matplotlib.pyplot as plt

plt.gray()

for i in range(4):

plt.matshow(digits.images[i])

Page58 df = pd.DataFrame(digits.data)



df.head()

df['target'] = digits.target

df[0:12]

X = df.drop('target',axis='columns')

y = df.target

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2)

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n\_estimators=20)

model.fit(X\_train, y\_train)

model.score(X\_test, y\_test)

y\_predicted = model.predict(X\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_predicted)

cm

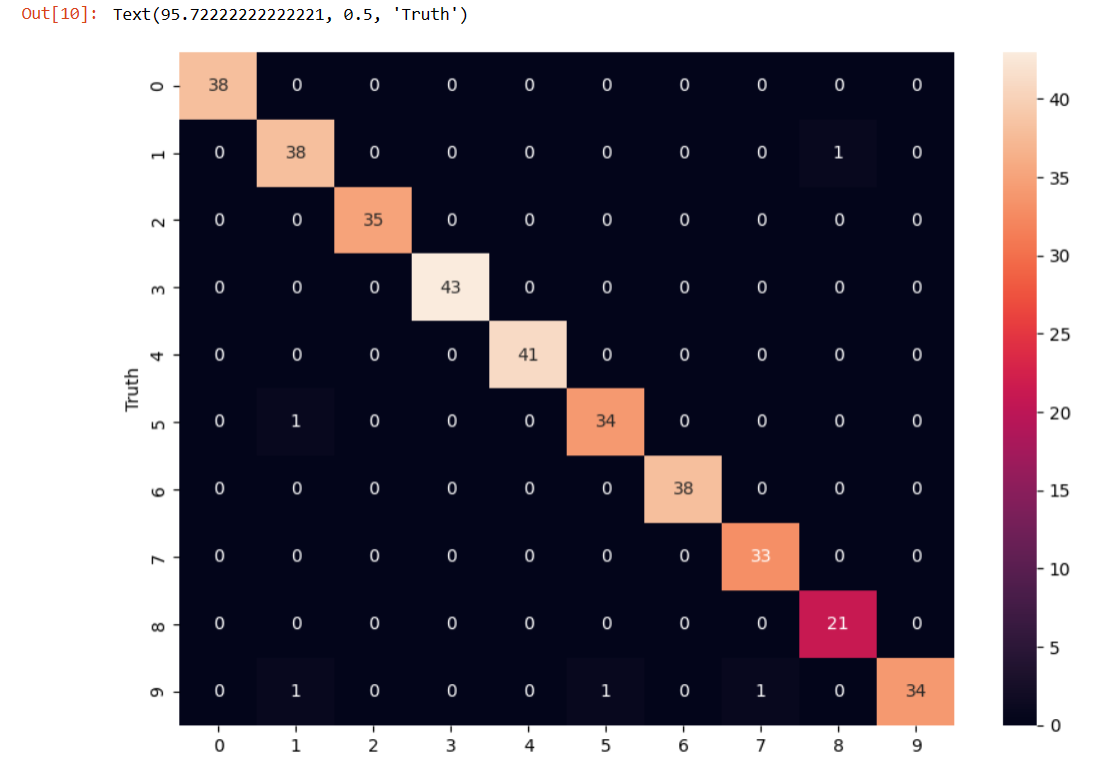
%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sn

plt.figure(figsize=(10,7)) Page59

sn.heatmap(cm, annot=True)

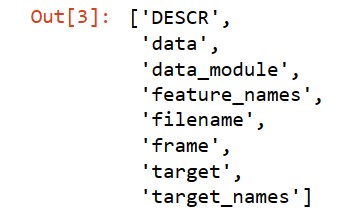
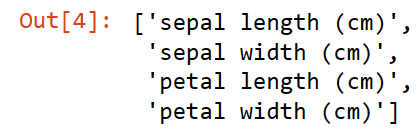


plt.xlabel('Predicted')

plt.ylabel('Truth')

**RESULT:**

Thus the program to implement random forest model was implemented and executed successfully. Page60



**Ex.No:7 Implement SVM Model**

**AIM:**

**ALGORITHM**

From sklearn datasets import load\_iris. Display the feature names from load\_iris Import pyplot from from matplotlib

Find the sepal length and sepal width from the given dataset Find the petal length and petal width from the trained dataset

**PROGRAM**

import pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

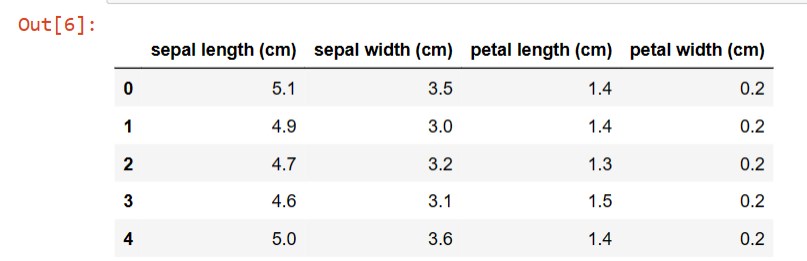
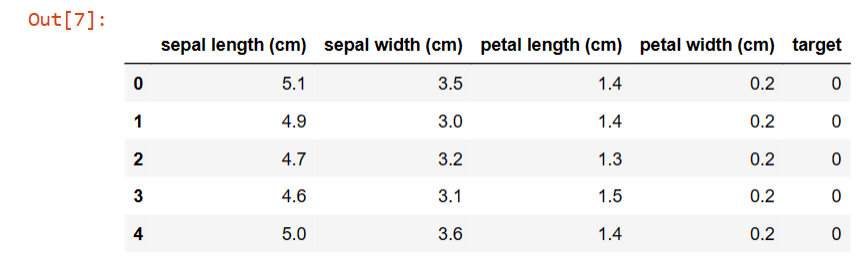
dir(iris)

iris.feature\_names

df=pd.DataFrame(iris.data, columns=iris.feature\_names)

df.head()

Page61



df['target']=iris.target

df.head()

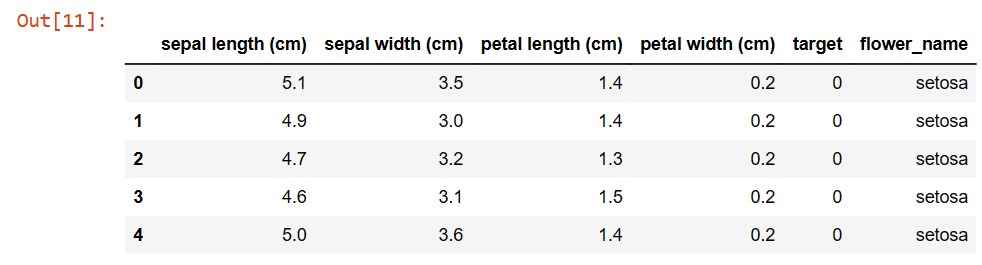
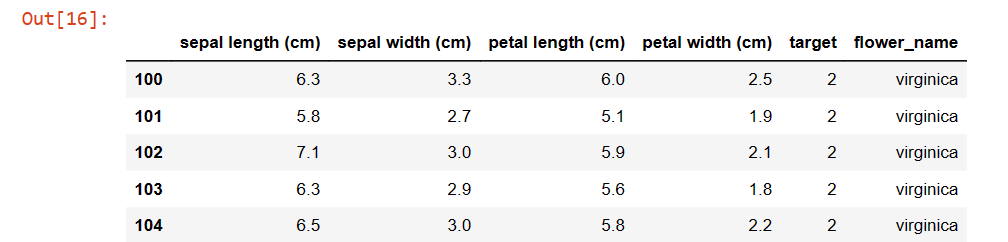
iris.target\_names

df[df.target==2].head

df['flower\_name']=df.target.apply(lambda x:iris.target\_names[x])

df.head()

Page62



from matplotlib import pyplot as plt

%matplotlib inline

df0=df[df.target==0]

df1=df[df.target==1]

df2=df[df.target==2]

df2.head()

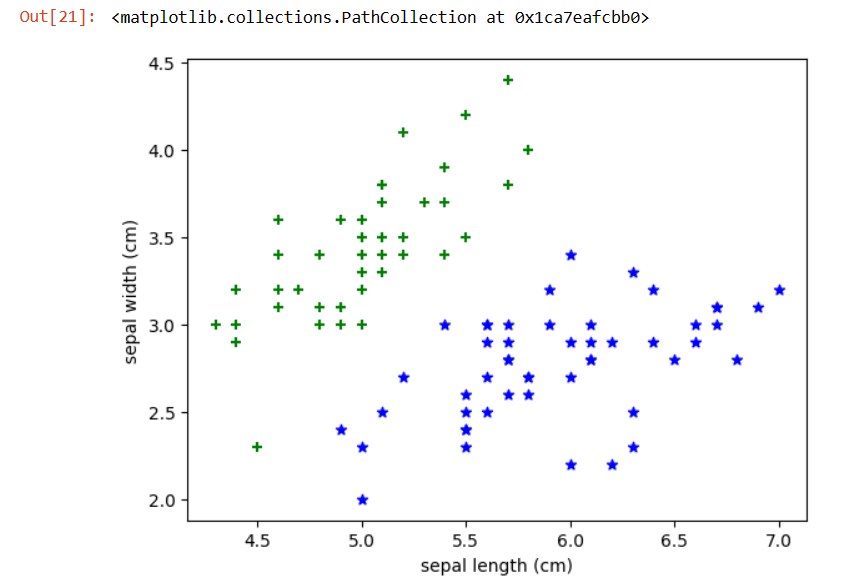
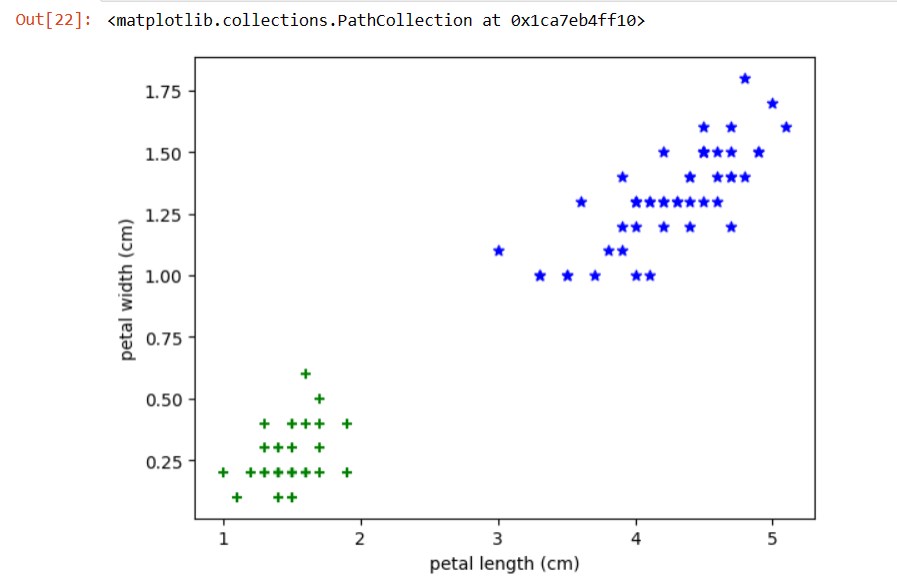
plt.xlabel('sepal length (cm)')

plt.ylabel('sepal width (cm)')

plt.scatter(df0['sepal length (cm)'],df0['sepal width (cm)'],color='green',marker='+')

plt.scatter(df1['sepal length (cm)'],df1['sepal width (cm)'],color='blue',marker='\*')

Page63



plt.xlabel('petal length (cm)')

plt.ylabel('petal width (cm)')

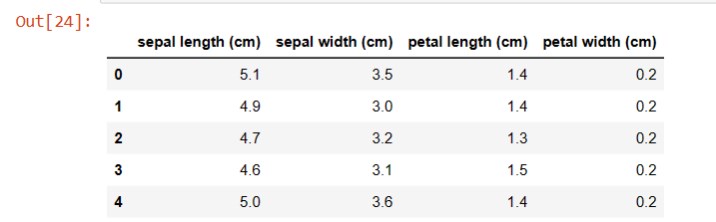
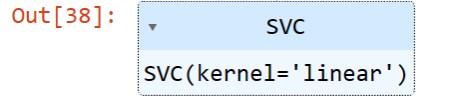
plt.scatter(df0['petal length (cm)'],df0['petal width (cm)'],color='green',marker='+')

plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue',marker='\*')

from sklearn.model\_selection import train\_test\_split

x = df.drop(['target','flower\_name'], axis='columns')

x.head() Page64



y=df.target

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x,y,test\_size=0.2)

len(x\_train)

len(x\_test)

from sklearn.svm import SVC

model = SVC(kernel='linear')

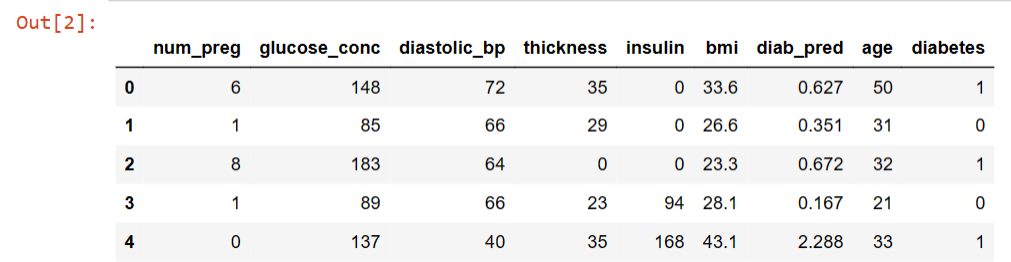
model.fit(x\_train, y\_train)

model.score(x\_test, y\_test)

**RESULT:**

Thus the program to implement SVM model was implemented and executedage65 successfully.

P



**Ex.No:8 Implement Ensembling Techniques(Bagging)**

**AIM:**

**ALGORITHM**

Import the panda’s library

Read the dataset from the path “C:/python/prima.csv”

Display the first five rows from the dataframe using head() function. Returns the number of missing values in the dataset using isnull.sum() function. Preprocess the given dataset using Standard scalar function

Find the cross value score using Decision tree classifier.

**PROGRAM**

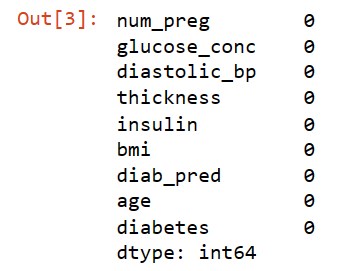
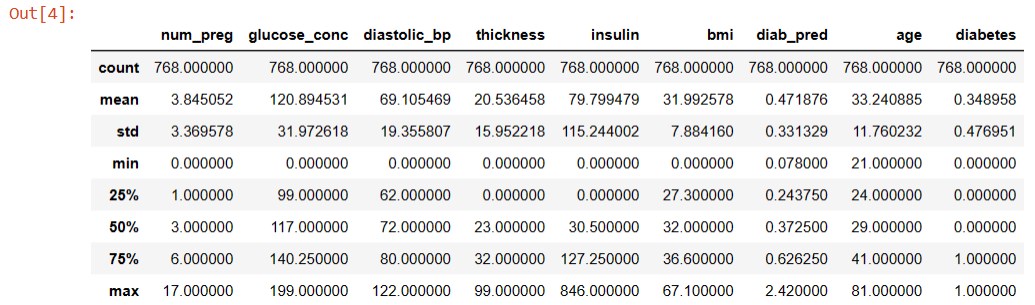
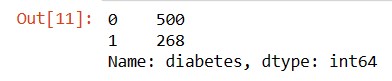
import pandas as pd

df = pd.read\_csv("C:/python/prima.csv")

df.head()

df.isnull().sum()

Page66



df.describe()

df.diabetes.value\_counts()

X = df.drop("diabetes",axis="columns")

y = df.diabetes

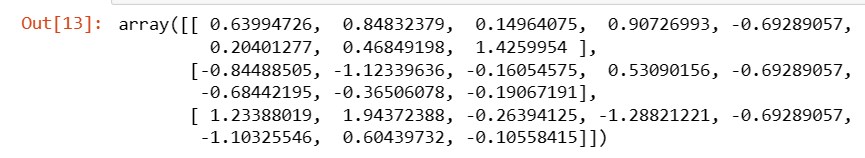
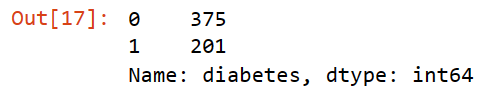
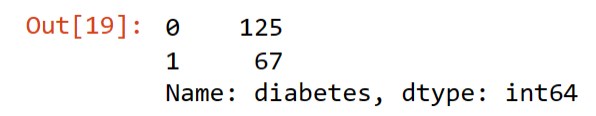
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_scaled[:3]

Page67



from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, stratify=y, random\_state=10)

X\_train.shape

X\_test.shape

y\_train.value\_counts()

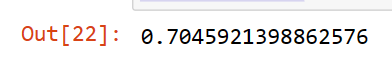
201/375

y\_test.value\_counts()

67/125

Page68

from sklearn.model\_selection import cross\_val\_score



from sklearn.tree import DecisionTreeClassifier

scores = cross\_val\_score(DecisionTreeClassifier(), X, y, cv=5)

scores

scores.mean()

from sklearn.ensemble import BaggingClassifier

bag\_model = BaggingClassifier(

base\_estimator=DecisionTreeClassifier(),

n\_estimators=100,

max\_samples=0.8,

oob\_score=True,

random\_state=0

)

bag\_model.fit(X\_train, y\_train)

bag\_model.oob\_score\_

bag\_model.score(X\_test, y\_test)

bag\_model = BaggingClassifier( Page69



base\_estimator=DecisionTreeClassifier(),

n\_estimators=100,

max\_samples=0.8,

oob\_score=True,

random\_state=0

)

scores = cross\_val\_score(bag\_model, X, y, cv=5)

scores

scores.mean()

**RESULT:**

Thus the program to implement ensembling techniques was implemented and

executed successfully.

Page70

**Ex.No:9 Implement Clustering Algorithms(KMeans)**

**AIM:**

Import MinMaxScaler from sklearn preprocessing From sklearn datasets import load\_iris

Display the first five rows of the dataset using head function

Apply Kmeans to the given dataset and find the septal length, septal width and petal length and petal width

**PROGRAM**

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

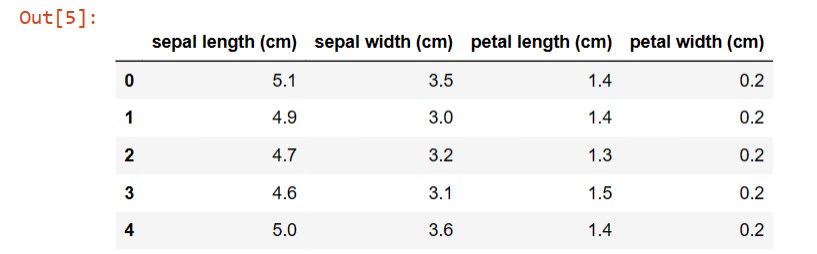
from sklearn.datasets import load\_iris

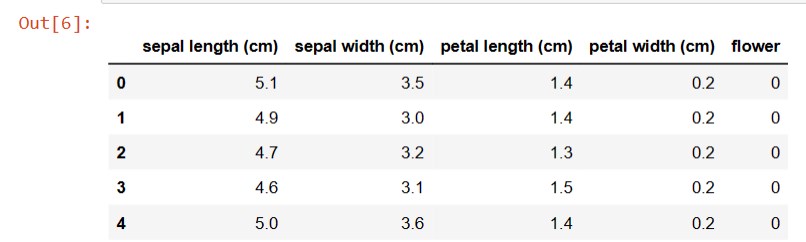
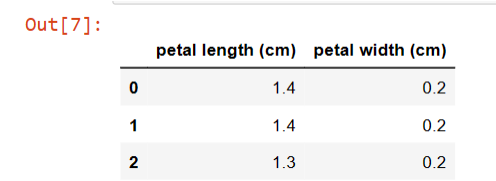
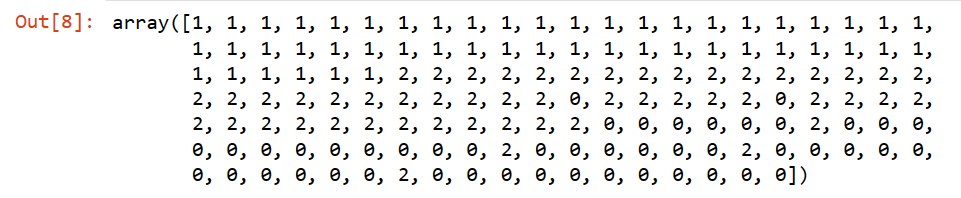
%matplotlib inline

iris = load\_iris()

df = pd.DataFrame(iris.data,columns=iris.feature\_names)

df.head()

Page71



df['flower'] = iris.target

df.head()

df.drop(['sepal length (cm)', 'sepal width (cm)', 'flower'],axis='columns',inplace=True)

df.head(3)

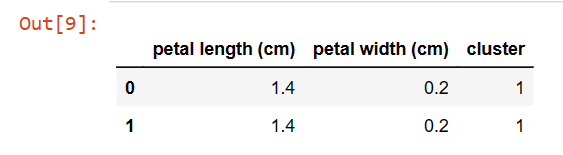
km = KMeans(n\_clusters=3)

yp = km.fit\_predict(df)

yp

df['cluster'] = yp

Page72 df.head(2)



df.cluster.unique()

df1 = df[df.cluster==0]

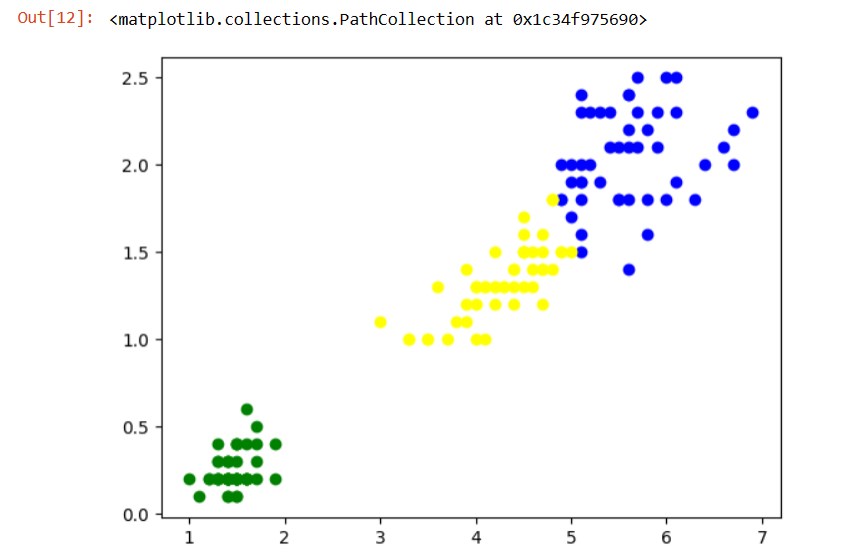
df2 = df[df.cluster==1]

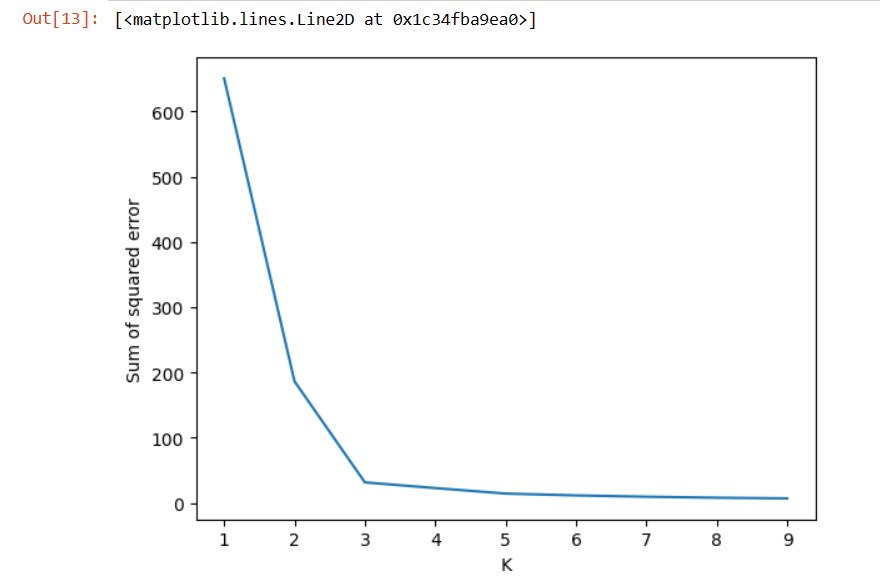
df3 = df[df.cluster==2]

plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue')

plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='green')

plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='yellow')

Page73 sse = []



k\_rng = range(1,10)

for k in k\_rng:

km = KMeans(n\_clusters=k)

km.fit(df)

sse.append(km.inertia\_)

plt.xlabel('K')

plt.ylabel('Sum of squared error')

plt.plot(k\_rng,sse)

**RESULT:**

Thus the program to implement clustering algorithm was implemented and executed

successfully.

Page74

**Ex.No:11 Build simple Neural Network**

**AIM:**

**ALGORITHM**

Define the activation function

Train the input values and and obtain the output from the given dataset. Test the given dataset from the output obtained from the given dataset Obtain the forward and Backward pass from the trained dataset

**PROGRAM**

# importing dependancies

import numpy as np

# The activation function

def activation(x):

return 1 / (1 + np.exp(-x))

weights = np.random.uniform(-1,1,size = (2, 1))

training\_inputs = np.array([[0, 0, 1, 1, 0, 1]]).reshape(3, 2)

training\_outputs = np.array([[0, 1, 1]]).reshape(3,1)

for i in range(15000):

# forward pass

dot\_product = np.dot(training\_inputs, weights)

output = activation(dot\_product)

# backward pass.

temp2 = -(training\_outputs - output) \* output \* (1 - output) Page75



adj = np.dot(training\_inputs.transpose(), temp2)

# 0.5 is the learning rate.

weights = weights - 0.5 \* adj

# The testing set

test\_input = np.array([1, 0])

test\_output = activation(np.dot(test\_input, weights))

# OR of 1, 0 is 1

print(test\_output)

**OUTPUT:**

**RESULT:**

Thus the program to implement neural network was implemented and executed

successfully.

Page76

**Ex.No:12 Build Deep Learning Neural Network model**

**AIM:**

**ALGORITHM**

Load data from the test file using import loadtxt

Import sequential from tensorflow Import Dense from tensor flow

Load data from from the test file from the path 'C:/python/pima-indians-diabetes.csv', delimiter=','

Split the given dataset into input and output variables. Fit the keras model on the given dataset.

**PROGRAM**

from numpy import loadtxt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

dataset = loadtxt('C:/python/pima-indians-diabetes.csv', delimiter=',')

# split into input (X) and output (y) variables

X = dataset[:,0:8]

y = dataset[:,8]

# define the keras model

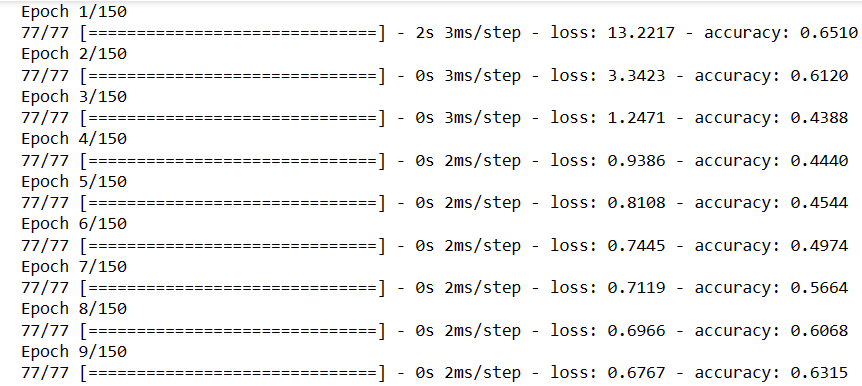
model = Sequential()

model.add(Dense(12, input\_shape=(8,), activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

Page77 # compile the keras model



model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the keras model on the dataset

model.fit(X, y, epochs=150, batch\_size=10)

# evaluate the keras model

\_, accuracy = model.evaluate(X, y)

print('Accuracy: %.2f' % (accuracy\*100))

model.fit(X, y, epochs=150, batch\_size=10, verbose=0)

# evaluate the keras model

\_, accuracy = model.evaluate(X, y, verbose=0)

# make probability predictions with the model

predictions = model.predict(X)

# round predictions

rounded = [round(x[0]) for x in predictions]

# make class predictions with the model

predictions = (model.predict(X) > 0.5).astype(int) Page78



from numpy import loadtxt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# load the dataset

dataset = loadtxt('C:/python/pima-indians-diabetes.csv', delimiter=',')

# split into input (X) and output (y) variables

X = dataset[:,0:8]

y = dataset[:,8]

# define the keras model

model = Sequential()

model.add(Dense(12, input\_shape=(8,), activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# compile the keras model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the keras model on the dataset

model.fit(X, y, epochs=150, batch\_size=10, verbose=0)

# make class predictions with the model

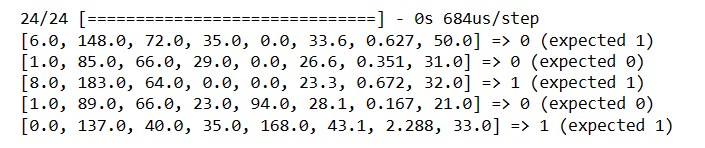
predictions = (model.predict(X) > 0.5).astype(int)

# summarize the first 5 cases

for i in range(5):

print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))

Page79



**RESULT:**

Thus the program to implement deep learning neural network was implemented and

executed successfully.

Page80