||Jai Sri Gurudev||

Sri AdichunchanagiriShikshana Trust(R)

SJB Institute of Technology

No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road
Kengeri, Bangalore – 560 060

Department of Information Science and Engineering

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY
[18CSL76]

VII SEMESTER – B. E

Staff Name:	**Dr.Rohini T V and Vishruth B Gowda**		
Section:	**A & B**	**Batch:**	**A1, A2, A3, B1, B2 and B3**

PREFACE

Artificial intelligence and machine learning deals with design of machines which are capable of learning and performing tasks ranging from ordinary category to expert category.

The current manual is prepared to equip students with information required to conduct experiments necessary to understand and implement artificial intelligence and machine learning concepts like A*, Candidate elimination, naïve bayes etc. The manual gives necessary information about basic concepts and techniques which could be applied to various domains for getting optimal solution.

This lab also helps student to develop some requisite knowledge to carry out final year project in the area of machine learning, data science, CNN etc. Overall it gives an exposure on hands-on aspects of the artificial intelligence and machine learning discipline.

SJB INTITUTE OF TECHNOLOGY

Institution’s Vision

To become a recognized technical education centre with a global perspective.

Institution’s Mission

To provide learning opportunities that foster students ethical values, intelligent development in science & technology and social responsibility so that they become sensible and contributing members of the society.

Department of Information Science and Engineering

Department Vision

We envision our department as a catalyst for developing educated, engaged and employable individuals whose collective energy will be the driving force for prosperity and the quality of life in our diverse world.

Department Mission

Our mission is to provide quality technical education in the field of information technology and to strive for excellence in the education by developing and sharpening the intellectual and human potential for good industry and community.

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

Graduates will -

- Possess expertise in problem solving, design and analysis, technical skills for a fruitful career accomplishing professional and social ethics with exposure to modern designing tools and technologies in Information Science and Engineering.
- Excel in communication, teamwork and multipledomains related to engineering issues accomplishing social responsibilities and management skills.
- Outclass in competitive environment through certification courses, gaining leadership qualities and progressive research to become successful entrepreneurs.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

Graduates will be able to -

1. **PSO1:** Apply the Knowledge of Information Science to develop software solutions in current research trends and technology.
2. **PSO2:** Create Social awareness & environmental wisdom along with ethical responsibility to lead a successful career and sustain passion using optimal resources to become an Entrepreneur.

PROGRAM OUTCOMES-PO’s

Engineering graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling 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

University Syllabus

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY
(Effective from the academic year 2021 -2022)
SEMESTER – VII

Course Code 18CSL76	**CIE Marks** 40
Number of Contact Hours/Week 0:0:2	**SEE Marks** 60
Total Number of Lab Contact Hours 36	**Exam Hours** 03

Credits – 2

Course Learning Objectives: This course (18CSL76) will enable students to:
• Implement and evaluate AI and ML algorithms in and Python programming language.

Descriptions (if any):
Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Programs List:

1. Implement **A* Search algorithm**.
2. Implement **AO* Search algorithm**.
3. For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination algorithm** to output a description of the set of all hypotheses consistent with the training examples.
4. Write a program to demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.
5. Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.
6. Write a program to implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.
7. Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set for clustering using **k-Means algorithm**. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.
8. Write a program to implement **k-Nearest Neighbour algorithm** to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.
9. Implement the non-parametric **Locally Weighted Regression algorithm** in order to fit data points. Select appropriate data set for your experiment and draw graphs

Laboratory Outcomes: The student should be able to:

- Implement and demonstrate AI and ML algorithms.
- Evaluate different algorithms.

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.

- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution *(Courseed to change in accoradance with university regulations)*
  - q) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 =100 Marks
  - r) For laboratories having PART A and PART B
    - i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
    - ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

COURSE OUTCOMES

On successful completion of this course students will be able to,

CO's	COURSE OUTCOMES	PO's and PSO's MAPPING
CO1	Explore various python libraries useful for real time application and apply appropriate data sets to the ML algorithm	PO1, PO4,PO5 / PSO1
CO2	Understand the implementation procedure for the machine learning and artificial intelligence algorithms.	PO1, PO2 / PSO1
CO3	Identify, apply and evaluate ML algorithms to solve real world problems.	PO1, PO2,PO3,PO4, PO5 / PSO1

General Instructions for the Laboratory

Do's

- It is mandatory for all the students to attend all practical classes &amp; complete the experiments as per syllabus.
- Students should strictly follow the lab timings, dress code with Apron &amp; ID cards.
- Should maintain a neat observation book.
- Study the theory and logic before executing the program.
- Submit the completed lab records of executed programs and update the index book in every lab session.
- Should prepare for viva questions regularly.
- Handle the computer systems carefully.
- Maintain discipline and silence in the lab.

Don'ts

- Should not take Bags and Mobile phones into the Laboratory.
- Do not wear footwear inside the Laboratory
- Systems &amp; Components should be handled carefully failing to which penalty will be imposed.
- Do not switch off the system abruptly.
- Should not chew gum or eat in the lab.

LAB INDEX SHEET

1. Implement **A* Search algorithm**.

```
def aStarAlgo(start_node, stop_node):

        open_set = set(start_node)
        print('open_set',open_set)
        closed_set = set()
        print('set',set)
        g = {} #store distance from starting node
        parents = {}# parents contains an adjacency map of all nodes
        print('parents',parents)

        #ditance of starting node from itself is zero
        g[start_node] = 0
        #start_node is root node i.e it has no parent nodes
        #so start_node is set to its own parent node
        parents[start_node] = start_node
        print('Start_node',start_node)


        while len(open_set) > 0:
            print('open_set1',open_set)
            n = None
            print('n',n)

            #node with lowest f() is found
            for v in open_set:
                if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

                    print('g[v]',g[v])
                    print('heuristic(v)',heuristic(v))
                    #print('g[n]',g[n])
                    #print('heuristic(n)',heuristic(n))
                    n = v
                    print('v',v)


            if n == stop_node or Graph_nodes[n] == None:
                print('n1',n)
                pass
            else:
                for (m, weight) in get_neighbors(n):
                    print('m',m)
                    print('weight',weight)
                    print('get_neighbors(n)',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:
                    print('open_set',open_set)
                    print('closed_set',closed_set)
                    open_set.add(m)
                    parents[m] = n
                    print('n1',n)
                    g[m] = g[n] + weight
                    print('g[m]',g[m])
                    print('g[n]',g[n])
                    print('weight',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:
                        print('g1[m]',g[m])
                        #update g(m)
                        g[m] = g[n] + weight
                        #change parent of m to n
                        parents[m] = n
                        print('n2',n)
                        #if m in closed set,remove and add to open
                        if m in closed_set:
                            print('m',m)
                            closed_set.remove(m)
                            open_set.add(m)
                            print('open_set.add(m)',open_set.add(m))

        if n == None:
            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:
            print('n3',n)
            path = []
            print('path',path)

            while parents[n] != n:
                print('n4',n)
                path.append(n)
                n = parents[n]
                print('n5',n)
                print('parents[n]',parents[n])

            path.append(start_node)
```

```
                path.reverse()
                print('path.reverse()',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(' closed_set.add(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:
        print('v1',v)
        return Graph_nodes[v]
        print(Graph_nodes[v])
    else:
        return None
#for simplicity we ll consider heuristic distances given
#and this function returns heuristic distance for all nodes
def heuristic(n):
        H_dist = {
            'S': 7,
            'A': 6,
            'B': 2,
            'C': 1,
            'D': 0,
        }

        return H_dist[n]
        print('H_dist[n]',H_dist[n])

#Describe your graph here
Graph_nodes = {
    'S': [('A', 1), ('B', 4)],
    'A': [('B', 2), ('D', 12)],
    'B': [('C', 2)],
    'C': [('D', 3)],
}
aStarAlgo('S', 'D')
```

Output

```
open_set {'S'}
set <class 'set'>
parents {}
Start_node S
open_set1 {'S'}
n None
g[v] 0
heuristic(v) 7
v S
v1 S
m A
weight 1
v1 S
get_neighbors(n) [('A', 1), ('B', 4)]
open_set {'S'}
closed_set set()
n1 S
g[m] 1
g[n] 0
weight 1
m B
weight 4
v1 S
get_neighbors(n) [('A', 1), ('B', 4)]
open_set {'A', 'S'}
closed_set set()
n1 S
g[m] 4
g[n] 0
weight 4
 closed_set.add(n) None
open_set1 {'A', 'B'}
n None
g[v] 1
heuristic(v) 6
v A
g[v] 4
heuristic(v) 2
v B
v1 B
```

```
m C
weight 2
v1 B
get_neighbors(n) [('C', 2)]
open_set {'A', 'B'}
closed_set {'S'}
n1 B
g[m] 6
g[n] 4
weight 2
 closed_set.add(n) None
open_set1 {'A', 'C'}
n None
g[v] 1
heuristic(v) 6
v A
v1 A
m B
weight 2
v1 A
get_neighbors(n) [('B', 2), ('D', 12)]
g1[m] 4
n2 A
m B
open_set.add(m) None
m D
weight 12
v1 A
get_neighbors(n) [('B', 2), ('D', 12)]
open_set {'A', 'B', 'C'}
closed_set {'S'}
n1 A
g[m] 13
g[n] 1
weight 12
 closed_set.add(n) None
open_set1 {'C', 'D', 'B'}
n None
g[v] 6
heuristic(v) 1
v C
g[v] 3
heuristic(v) 2
v B
v1 B
m C
```

```
weight 2
v1 B
get_neighbors(n) [('C', 2)]
g1[m] 6
n2 B
 closed_set.add(n) None


open_set1 {'C', 'D'}
n None
g[v] 5
heuristic(v) 1
v C
v1 C
m D
weight 3
v1 C
get_neighbors(n) [('D', 3)]
g1[m] 13
n2 C
 closed_set.add(n) None
open_set1 {'D'}
n None
g[v] 8
heuristic(v) 0
v D
n1 D
n3 D
path []
n4 D
n5 C
parents[n] B
n4 C
n5 B
parents[n] A
n4 B
n5 A
parents[n] S
n4 A
n5 S
parents[n] S
path.reverse() None
Path found: ['D', 'C', 'B', 'A', 'S']

 ['D', 'C', 'B', 'A', 'S']
```

2. Implement **AO* Search algorithm**.

```
class Graph:
   def __init__(self, graph, heuristicNodeList, startNode):  #instantiate graph object with graph topology,
heuristic values, start node

        self.graph = graph
        self.H=heuristicNodeList
        self.start=startNode
        self.parent={}
        self.status={}
        self.solutionGraph={}

   def applyAOStar(self):        # starts a recursive AO* algorithm
        self.aoStar(self.start, False)

   def getNeighbors(self, v):     # gets the Neighbors of a given node
        return self.graph.get(v,'')

   def getStatus(self,v):        # return the status of a given node
        return self.status.get(v,0)

   def setStatus(self,v, val):    # set the status of a given node
        self.status[v]=val

   def getHeuristicNodeValue(self, n):
        return self.H.get(n,0)     # always return the heuristic value of a given node

   def setHeuristicNodeValue(self, n, value):
        self.H[n]=value           # set the revised heuristic value of a given node


   def printSolution(self):
        print("FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START
        NODE:",self.start)
        print("-----------------------------------------------------------")
        print(self.solutionGraph)
        print("-----------------------------------------------------------")

   def computeMinimumCostChildNodes(self, v):  # Computes the Minimum Cost of child nodes of a
                                                                given node v
        minimumCost=0
        costToChildNodeListDict={}
        costToChildNodeListDict[minimumCost]=[]
        flag=True
        for nodeInfoTupleList in self.getNeighbors(v):  # iterate over all the set of child node/s
        cost=0
        nodeList=[]
        for c, weight in nodeInfoTupleList:
            cost=cost+self.getHeuristicNodeValue(c)+weight
            nodeList.append(c)

        if flag==True:                      # initialize Minimum Cost with the cost of first set of child node/s
            minimumCost=cost
            costToChildNodeListDict[minimumCost]=nodeList     # set the Minimum Cost child node/s
            flag=False
        else:                               # checking the Minimum Cost nodes with the current Minimum Cost
            if minimumCost>cost:
                minimumCost=cost
                costToChildNodeListDict[minimumCost]=nodeList # set the Minimum Cost child node/s


    return minimumCost, costToChildNodeListDict[minimumCost]   # return Minimum Cost and
                                                                 Minimum Cost child node/s


  def aoStar(self, v, backTracking):     # AO* algorithm for a start node and backTracking status flag

    print("HEURISTIC VALUES  :", self.H)
    print("SOLUTION GRAPH    :", self.solutionGraph)
    print("PROCESSING NODE   :", v)
    print("-----------------------------------------------------------------------------------------")

    if self.getStatus(v) >= 0:        # if status node v >= 0, compute Minimum Cost nodes of v
        minimumCost, childNodeList = self.computeMinimumCostChildNodes(v)
        self.setHeuristicNodeValue(v, minimumCost)
        self.setStatus(v,len(childNodeList))

        solved=True                   # check the Minimum Cost nodes of v are solved
        for childNode in childNodeList:
            self.parent[childNode]=v
            if self.getStatus(childNode)!=-1:
                solved=solved & False

        if solved==True:              # if the Minimum Cost nodes of v are solved, set the current node
                                          status as solved(-1)
            self.setStatus(v,-1)
            self.solutionGraph[v]=childNodeList        # update the solution graph with the solved nodes
                                                       which may be a part of solution

        if v!=self.start:           # check the current node is the start node for backtracking the current
                                     node value
            self.aoStar(self.parent[v], True)   # backtracking the current node value with backtracking
                                                 status set to true
        if backTracking==False:     # check the current call is not for backtracking
           for childNode in childNodeList:   # for each Minimum Cost child node
              self.setStatus(childNode,0)   # set the status of child node to 0(needs exploration)
              self.aoStar(childNode, False) # Minimum Cost child node is further explored with
                                                      backtracking status as false

h1 = {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1, 'T': 3}
graph1 = {
   'A': [[('B', 1), ('C', 1)], [('D', 1)]],
   'B': [[('G', 1)], [('H', 1)]],
   'C': [[('J', 1)]],
   'D': [[('E', 1), ('F', 1)]],
   'G': [[('I', 1)]]
}
G1= Graph(graph1, h1, 'A')
G1.applyAOStar()
G1.printSolution()

h2 = {'A': 1, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5, 'H': 7}  # Heuristic values of Nodes
graph2 = {                                    # Graph of Nodes and Edges
   'A': [[('B', 1), ('C', 1)], [('D', 1)]],      # Neighbors of Node 'A', B, C & D with repective weights
   'B': [[('G', 1)], [('H', 1)]],               # Neighbors are included in a list of lists
   'D': [[('E', 1), ('F', 1)]]                  # Each sublist indicate a "OR" node or "AND" nodes
}

G2 = Graph(graph2, h2, 'A')                     # Instantiate Graph object with graph, heuristic values and start
Node
G2.applyAOStar()                               # Run the AO* algorithm
G2.printSolution()                             # Print the solution graph as output of the AO* algorithm search
```

Output

```
HEURISTIC VALUES  : {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H':
7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : A
-----------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : B
-----------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : A
```

```
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : G
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : B
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 10, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : A
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8,
'H': 7, 'I': 7, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {}
PROCESSING NODE   : I
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8,
'H': 7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': []}
PROCESSING NODE   : G
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1,
'H': 7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I']}
PROCESSING NODE   : B
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 12, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1,
'H': 7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G']}
PROCESSING NODE   : A
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H':
7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G']}
PROCESSING NODE   : C
-------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H':
7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G']}
PROCESSING NODE   : A
-------------------------------------------------------------------------------
-------
```

```
HEURISTIC VALUES  : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H':
7, 'I': 0, 'J': 1, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G']}
PROCESSING NODE   : J
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H':
7, 'I': 0, 'J': 0, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G'], 'J': []}
PROCESSING NODE   : C
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 6, 'B': 2, 'C': 1, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H':
7, 'I': 0, 'J': 0, 'T': 3}
SOLUTION GRAPH    : {'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J']}
PROCESSING NODE   : A
--------------------------------------------------------------------------------
-------
FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: A
------------------------------------------------------------
{'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J'], 'A': ['B', 'C']}
------------------------------------------------------------
HEURISTIC VALUES  : {'A': 1, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {}
PROCESSING NODE   : A
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {}
PROCESSING NODE   : D
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {}
PROCESSING NODE   : A
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 4, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {}
PROCESSING NODE   : E
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 11, 'B': 6, 'C': 12, 'D': 10, 'E': 0, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {'E': []}
PROCESSING NODE   : D
--------------------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 11, 'B': 6, 'C': 12, 'D': 6, 'E': 0, 'F': 4, 'G': 5,
'H': 7}
SOLUTION GRAPH    : {'E': []}
PROCESSING NODE   : A
```

```
----------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 7, 'B': 6, 'C': 12, 'D': 6, 'E': 0, 'F': 4, 'G': 5, 'H': 
7}
SOLUTION GRAPH    : {'E': []}
PROCESSING NODE   : F
----------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 7, 'B': 6, 'C': 12, 'D': 6, 'E': 0, 'F': 0, 'G': 5, 'H': 
7}
SOLUTION GRAPH    : {'E': [], 'F': []}
PROCESSING NODE   : D
----------------------------------------------------------------------
-------
HEURISTIC VALUES  : {'A': 7, 'B': 6, 'C': 12, 'D': 2, 'E': 0, 'F': 0, 'G': 5, 'H': 
7}
SOLUTION GRAPH    : {'E': [], 'F': [], 'D': ['E', 'F']}
PROCESSING NODE   : A
----------------------------------------------------------------------
-------
FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: A
------------------------------------------------------------
{'E': [], 'F': [], 'D': ['E', 'F'], 'A': ['D']}
------------------------------------------------------------
```

3. For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination algorithm** to output a description of the set of all hypotheses consistent with the training examples.

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('datasetprog3.csv'))
print(data)

concepts = np.array(data.iloc[:,0:-1])
print(concepts)

target = np.array(data.iloc[:,-1])
print(target)

def learn(concepts, target):
    print('Most Specific Hypothesis:\n')
    specific_h = ["@" for i in range(6)]
    print(specific_h)
    for i in range(len(target)):
        if target[i] == 'Yes':
            specific_h = concepts[i].copy()
            break

    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]

    print('\nMost General Hypothesis:\n')
    for g in general_h:
        print(g)

    for i, h in enumerate(concepts):
        print('\nExample',i,': ',h,' Target:',target[i])
        if target[i] == "Yes":
            for x in range(len(specific_h)):

                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'

        if target[i] == "No":
            for x in range(len(specific_h)):

                if h[x] != specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print('S:\n',specific_h)
        print('G:')
        for g in general_h:
            print(g)

    indices = [i for i,val in enumerate(general_h)
                    if val == ['?', '?', '?', '?', '?', '?']]
    print(indices)

    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])

    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final S", s_final, sep="\n")
print("Final G:\n")
for g in g_final:
print(g)
```

Output

```
      Sky Air Temperature Humidity  Wind Water Weather Forecast EnjoySport
      0  Sunny          Warm   Normal  Strong  Warm     Same       Yes
      1  Sunny          Warm    High  Strong  Warm     Same       Yes
      2  Rainy         Cold    High  Strong  Warm    Change     No
      3  Sunny          Warm    High  Strong  Cool    Change     Yes


 [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
['Sunny'  'Warm'  'High'  'Strong'  'Cool'  'Change']]  ['Yes'

'Yes' 'No' 'Yes']


Most   Specific   Hypothesis:  ['@',

'@', '@', '@', '@','@']

Most General Hypothesis:

['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
```

```
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']

Example 0 : ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same'] Target: Yes S:
 ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same'] G:
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']

Example 1 : ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same'] Target: Yes S:
 ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same'] G:
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']

Example 2 : ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change'] Target: No S:
 ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same'] G:
['Sunny', '?', '?', '?', '?', '?']
['?', 'Warm', '?', '?', '?','?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', 'Same']

Example 3 : ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change'] Target: Yes S:
 ['Sunny' 'Warm' '?' 'Strong' '?' '?'] G:
['Sunny', '?', '?', '?', '?', '?']
['?', 'Warm', '?', '?', '?','?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?'] [2, 3,
4, 5]
 Final S:
 ['Sunny' 'Warm' '?' 'Strong' '?' '?']

 Final G:

 ['Sunny', '?', '?', '?', '?', '?']
 ['?', 'Warm', '?', '?', '?','?']
```

4. Write a program to demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

```
import csv
import math
import random
class Node():
    value= ""
    children = []
    def __init__(self, val, dictionary):
        self.value=val
        if(isinstance(dictionary,dict)):
            self.children=dictionary.keys()
def majorClass(attributes,data,target):
    freq={}
    index=attributes.index(target)
    for tuple in data:
        if tuple[index] in freq:
            freq[tuple[index]]+=1
        else:
            freq[tuple[index]]=1
    max=0
    major=""
    for key in freq.keys():
        if freq[key]>max:
            max=freq[key]
            major = key

    return major
def entropy(attributes,data,targetAttr):
    freq={}
    dataEntropy=0.0
    i=0
    for entry in attributes:
        if(targetAttr==entry):
            break
        i=i+1
    i=i-1
    for entry in data:
        if entry[i] in freq:
            freq[entry[i]]+=1.0
        else:
            freq[entry[i]]=1.0
    for freq in freq.values():
        dataEntropy+=(-freq/len(data))*math.log(freq/len(data),2)
    return dataEntropy
```

```
def info_gain(attributes,data,attr,targetAttr):
    freq={}
    subsetEntropy=0.0
    i=attributes.index(attr)

    for entry in data:
        if entry[i] in freq:
            freq[entry[i]]+=1.0
        else:
            freq[entry[i]]=1.0

    for val in freq.keys():
        valProb       = freq[val]/sum(freq.values())
        dataSubset    = [entry for entry in data if entry[i] == val]
        subsetEntropy += valProb * entropy(attributes, dataSubset, targetAttr)
    return(entropy(attributes, data, targetAttr) - subsetEntropy)

def attr_choose(data, attributes, target):
    best=attributes[0]
    maxGain=0;

    for attr in attributes:
            newGain = info_gain(attributes, data, attr, target)
            if newGain>maxGain:
                maxGain=newGain
                best=attr
    return best

def get_values(data, attributes, attr):
    index=attributes.index(attr)
    values=[]

    for entry in data:
        if entry[index] not in values:
            values.append(entry[index])
    return values

def get_data(data, attributes, best, val):
    new_data=[[]]
    index = attributes.index(best)
    for entry in data:
        if(entry[index] == val):
            newEntry = []
            for i in range(0,len(entry)):
                if(i != index):
                    newEntry.append(entry[i])
            new_data.append(newEntry)
    new_data.remove([])
```

```
        return new_data

    def build_tree(data, attributes, target):
        data = data[:]
        vals = [record[attributes.index(target)] for record in data]
        default = majorClass(attributes, data, target)
        if not data or (len(attributes) - 1) <= 0:
            return default
        elif vals.count(vals[0]) == len(vals):
            return vals[0]
        else:
            best = attr_choose(data, attributes, target)
            tree={best:{}}

            for val in get_values(data, attributes, best):
                new_data = get_data(data, attributes, best, val)
                newAttr = attributes[:]
                newAttr.remove(best)

                subtree= build_tree(new_data, newAttr, target)
                tree[best][val] = subtree
        return tree
    def execute_decision_tree():
        data = []
        with open("datasetprog3.csv") as tsv:
            for line in csv.reader(tsv):
                data.append(tuple(line))
            print("number of records:",len(data))

            attributes=['outlook','temp','humidity','wind','play']
            target=attributes[-1]
            acc = []
            training_set = [x for i, x in enumerate(data)]
            tree = build_tree(training_set, attributes, target)
            results = []
            test_set = [('rain','mild','high','strong')]
            for entry in test_set:
                tempDict = tree.copy()
                result = ""
                while(isinstance(tempDict, dict)):
                    root = Node(next(iter(tempDict)), tempDict[next(iter(tempDict))])
                    tempDict = tempDict[next(iter(tempDict))]
                    index = attributes.index(root.value)
                    value = entry[index]
                    if(value in tempDict.keys()):
                        child = Node(value, tempDict[value])
                        result = tempDict[value]
                        tempDict = tempDict[value]
                    else:
                        result = "Null"
                        break
                if result != "Null":
                    results.append(result == entry[-1])
            print(result)

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

Output

INPUT	**OUTPUT**
for the input Rain Mild High Strong	**Output 1:** **Number of records: 15** **No**
for the input Rain Cool Normal Weak	Output 2: Number of records: 15 Yes
for the input Overcast Hot Normal strong	Output 3: Number of records: 15 Null

5. Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.

```
import numpy as np
x=np.array(([2,9],[1,5],[3,6]),dtype=float)
y=np.array(([92],[86],[89]),dtype=float)
x=x/np.amax(x,axis=0)
y=y/100

def sigmoid(x):
    return 1/(1+np.exp(-x))
def derivatives_sigmoid(x):
    return x*(1-x)
epoch=7000
lr=0.1
inputlayer_neurons=2
hiddenlayer_neurons=3
output_neurons=1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
    hinp1=np.dot(x,wh)
    hinp=hinp1+bh
    hlayer_act=sigmoid(hinp)
    outinpl=np.dot(hlayer_act,wout)
    outinp=outinpl+bout
    output=sigmoid(outinp)
    EO=y-output
    outgrad=derivatives_sigmoid(output)
    d_output=EO*outgrad
    EH=d_output.dot(wout.T)
    hiddengrad=derivatives_sigmoid(hlayer_act)
    d_hiddenlayer=EH*hiddengrad
    wout+=hlayer_act.T.dot(d_output)*lr
    wh+=x.T.dot(d_hiddenlayer)*lr
print("input:\n"+str(x))
print("Actual output:\n"+str(y))
print("predicted output:\n",output)
```

Output

```
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
```

```
Actual output:
[[0.92]
[0.86]
[0.89]]

predicted output:
[[0.89441315]
[0.88223862]
[0.89305143]]
```

6. Write a program to implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

```
# Example of Naive Bayes implemented from Scratch in Python
# Online Resource: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

import csv
import random
import math

def loadCsv(filename):
  lines = csv.reader(open(filename, "r"))
  dataset = list(lines)
  for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
  return dataset

def splitDataset(dataset, splitRatio):
  trainSize = int(len(dataset) * splitRatio)
  trainSet = []
  copy = list(dataset)
  while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
  return [trainSet, copy]

def separateByClass(dataset):
  separated = {}
  for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
              separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
  return separated

def mean(numbers):
  return sum(numbers)/float(len(numbers))

def stdev(numbers):
  avg = mean(numbers)
  variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
  return math.sqrt(variance)

def summarize(dataset):
  summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
  del summaries[-1]
  return summaries
```

```
def summarizeByClass(dataset):
  separated = separateByClass(dataset)
  summaries = {}
  for classValue, instances in separated.items():
          summaries[classValue] = summarize(instances)
  return summaries

def calculateProbability(x, mean, stdev):
  exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
  return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

def calculateClassProbabilities(summaries, inputVector):
  probabilities = {}
  for classValue, classSummaries in summaries.items():
          probabilities[classValue] = 1
          for i in range(len(classSummaries)):
                  mean, stdev = classSummaries[i]
                  x = inputVector[i]
                  probabilities[classValue] *= calculateProbability(x, mean, stdev)
  return probabilities

def predict(summaries, inputVector):
  probabilities = calculateClassProbabilities(summaries, inputVector)
  bestLabel, bestProb = None, -1
  for classValue, probability in probabilities.items():
          if bestLabel is None or probability > bestProb:
                  bestProb = probability
                  bestLabel = classValue
  return bestLabel

def getPredictions(summaries, testSet):
  predictions = []
  for i in range(len(testSet)):
          result = predict(summaries, testSet[i])
          predictions.append(result)
  return predictions

def getAccuracy(testSet, predictions):
  correct = 0
  for i in range(len(testSet)):
          if testSet[i][-1] == predictions[i]:
                  correct += 1
  return (float(correct)/float(len(testSet))) * 100.0

def main():
  filename = 'pima-indians-diabetes-prog5.csv'
  dataset = loadCsv(filename)
```

```
    trainingSet=dataset
    testSet=loadCsv('pima-indians-diabetes-test-1-prog5.csv')
    print('Records in training data={1} and test data={2} rows'.format(len(dataset), len(trainingSet), len(testSet)))
    # prepare model
    summaries = summarizeByClass(trainingSet)
    # test model
    predictions = getPredictions(summaries, testSet)
    print(predictions)
    accuracy = getAccuracy(testSet, predictions)
    print("Accuracy:",accuracy,"%")

main()
```

Output

```
Records in training data=768 and test data=11 rows
[1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0]
Accuracy: 81.81818181818183 %
```

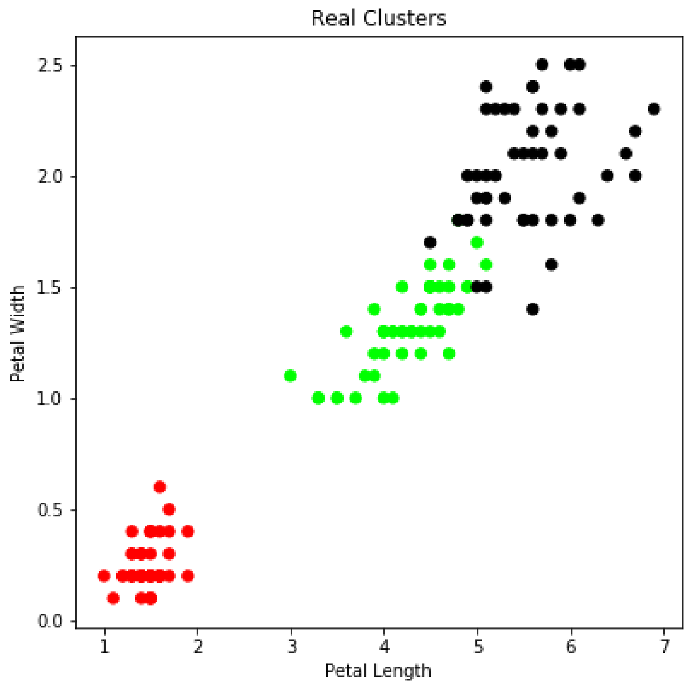
7. Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set for clustering using **k-Means algorithm**. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

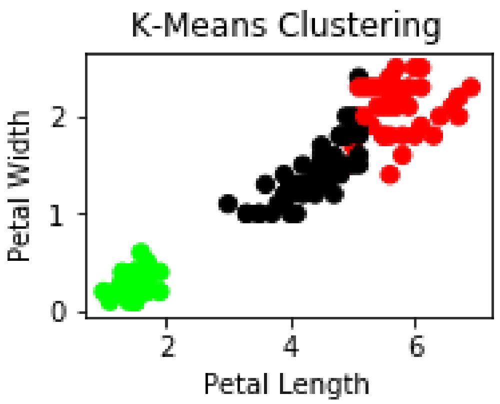
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np


# import some data to play with
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

# Build the K Means Model
model = KMeans(n_clusters=3)
# model.labels_ : Gives cluster no for which samples belongs to
model.fit(X)

# Visualise the clustering results
plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])

# Plot the Original Classifications using Petal features
plt.subplot(2, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()

# Plot the Models Classifications
plt.subplot(2, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()

# General EM for GMM
from sklearn import preprocessing
# transform your data such that its distribution will have a
# mean value 0 and standard deviation of 1.
```

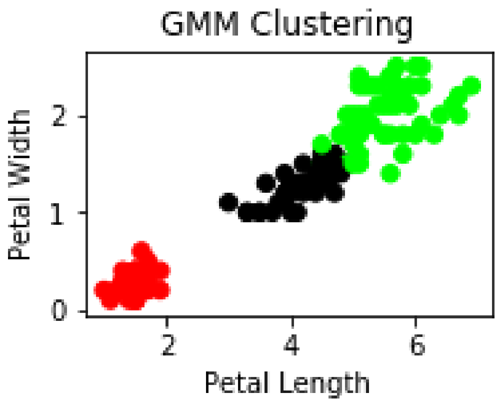
```
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)


gmm_y = gmm.predict(xs)
plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[gmm_y], s=40)
plt.title('GMM Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
print('Observation: The GMM using EM algorithm based clustering matched the true labels more closely than the Kmeans.')
```

Output

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
 n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
 random_state=None, tol=0.0001, verbose=0)
```







Observation: The GMM using EM algorithm based clustering matched the true labels more closely than the Kmeans.

8. Write a program to implement **k-Nearest Neighbour algorithm** to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

```
import numpy as np
from sklearn import preprocessing,cross_validation,neighbors
import pandas as pd
import matplotlib.pyplot as plt
import warnings

df=pd.read_csv("iris_prog9.csv")
print(df)
df.replace('setosa',1, inplace=True)
df.replace('versicolor',2, inplace=True)
df.replace('virginica',3, inplace=True)

#Missing Data Handling
df.replace('?',-9999,inplace=True)

#Define Attributes and Classes
X=np.array(df.drop(['species'],1))
Y=np.array(df['species'])

X_train,X_test,Y_train,Y_test= cross_validation.train_test_split(X,Y,test_size=0.8)


# Define the classifier using panda library

clf=neighbors.KNeighborsClassifier()

# Save the model with the fit method
clf.fit(X_train,Y_train)

# use the test set and calculate the accuracy of the model
accuracy=clf.score(X_test, Y_test)

print("Accurancy:")
print(accuracy)

print("-------------------------------------------------")
example_measures = np.array([[5.1,2.4,4.3,1.3],[4.9,3.0,1.4,0.2]])
example_measures = example_measures.reshape(2,-1)

prediction = clf.predict(example_measures)

print("Prediction")
print(prediction)
```

Output

```
sepal_length  sepal_width  petal_length  petal_width    species
0           5.1         3.5          1.4         0.2     setosa
1           4.9         3.0          1.4         0.2     setosa
2           4.7         3.2          1.3         0.2     setosa
3           4.6         3.1          1.5         0.2     setosa
4           5.0         3.6          1.4         0.2     setosa
5           5.4         3.9          1.7         0.4     setosa
6           4.6         3.4          1.4         0.3     setosa
7           5.0         3.4          1.5         0.2     setosa
8           4.4         2.9          1.4         0.2     setosa
9           4.9         3.1          1.5         0.1     setosa
10           5.4         3.7          1.5         0.2     setosa
11           4.8         3.4          1.6         0.2     setosa
12           4.8         3.0          1.4         0.1     setosa
13           4.3         3.0          1.1         0.1     setosa
14           5.8         4.0          1.2         0.2     setosa
15           5.7         4.4          1.5         0.4     setosa
16           5.4         3.9          1.3         0.4     setosa
17           5.1         3.5          1.4         0.3     setosa
18           5.7         3.8          1.7         0.3     setosa
19           5.1         3.8          1.5         0.3     setosa
20           5.4         3.4          1.7         0.2     setosa
21           5.1         3.7          1.5         0.4     setosa
22           4.6         3.6          1.0         0.2     setosa
23           5.1         3.3          1.7         0.5     setosa
24           4.8         3.4          1.9         0.2     setosa
25           5.0         3.0          1.6         0.2     setosa
26           5.0         3.4          1.6         0.4     setosa
27           5.2         3.5          1.5         0.2     setosa
28           5.2         3.4          1.4         0.2     setosa
29           4.7         3.2          1.6         0.2     setosa
..          ...         ...          ...         ...        ...
120           6.9         3.2          5.7         2.3  virginica
121           5.6         2.8          4.9         2.0  virginica
122           7.7         2.8          6.7         2.0  virginica
123           6.3         2.7          4.9         1.8  virginica
124           6.7         3.3          5.7         2.1  virginica
125           7.2         3.2          6.0         1.8  virginica
126           6.2         2.8          4.8         1.8  virginica
127           6.1         3.0          4.9         1.8  virginica
128           6.4         2.8          5.6         2.1  virginica
129           7.2         3.0          5.8         1.6  virginica
130           7.4         2.8          6.1         1.9  virginica
131           7.9         3.8          6.4         2.0  virginica
132           6.4         2.8          5.6         2.2  virginica
133           6.3         2.8          5.1         1.5  virginica
134           6.1         2.6          5.6         1.4  virginica
135           7.7         3.0          6.1         2.3  virginica
136           6.3         3.4          5.6         2.4  virginica
137           6.4         3.1          5.5         1.8  virginica
```

```
138          6.0          3.0          4.8          1.8  virginica
139          6.9          3.1          5.4          2.1  virginica
140          6.7          3.1          5.6          2.4  virginica
141          6.9          3.1          5.1          2.3  virginica
142          5.8          2.7          5.1          1.9  virginica
143          6.8          3.2          5.9          2.3  virginica
144          6.7          3.3          5.7          2.5  virginica
145          6.7          3.0          5.2          2.3  virginica
146          6.3          2.5          5.0          1.9  virginica
147          6.5          3.0          5.2          2.0  virginica
148          6.2          3.4          5.4          2.3  virginica
149          5.9          3.0          5.1          1.8  virginica

[150 rows x 5 columns]
Accurancy:
0.9583333333333334
--------------------------------------------------
Prediction
[2 1]
```

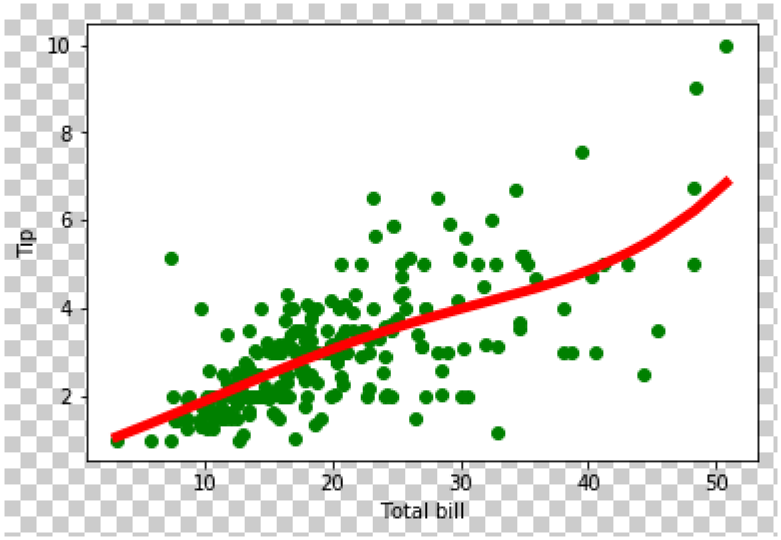
9. Implement the non-parametric **Locally Weighted Regression algorithm** in order to fit data points. Select appropriate data set for your experiment and draw graphs

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

def kernel(point,xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m))) # eye - identity matrix
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights
def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W

def localWeightRegression(xmat,ymat,k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred

def graphPlot(X,ypred):
    sortindex = X[:,1].argsort(0) #argsort - index of the smallest
    xsort = X[sortindex][:,0]
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.scatter(bill,tip, color='green')
    ax.plot(xsort[:,1],ypred[sortindex], color = 'red', linewidth=5)
    plt.xlabel('Total bill')
    plt.ylabel('Tip')
    plt.show();

# load data points
data = pd.read_csv('prog10_tips.csv')
bill = np.array(data.total_bill) # We use only Bill amount and Tips data
print (bill)
tip = np.array(data.tip)
mbill = np.mat(bill) # .mat will convert nd array is converted in 2D array
mtip = np.mat(tip)
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T)) # 244 rows, 2 cols
# increase k to get smooth curves
```

```
        ypred = localWeightRegression(X,mtip,9)
        graphPlot(X,ypred)
```

Output

```
[16.99 10.34 21.01 23.68 24.59 25.29  8.77 26.88 15.04 14.78 10.27 35.26
15.42 18.43 14.83 21.58 10.33 16.29 16.97 20.65 17.92 20.29 15.77 39.42
19.82 17.81 13.37 12.69 21.7  19.65  9.55 18.35 15.06 20.69 17.78 24.06
16.31 16.93 18.69 31.27 16.04 17.46 13.94  9.68 30.4  18.29 22.23 32.4
28.55 18.04 12.54 10.29 34.81  9.94 25.56 19.49 38.01 26.41 11.24 48.27
20.29 13.81 11.02 18.29 17.59 20.08 16.45  3.07 20.23 15.01 12.02 17.07
26.86 25.28 14.73 10.51 17.92 27.2  22.76 17.29 19.44 16.66 10.07 32.68
15.98 34.83 13.03 18.28 24.71 21.16 28.97 22.49  5.75 16.32 22.75 40.17
27.28 12.03 21.01 12.46 11.35 15.38 44.3  22.42 20.92 15.36 20.49 25.21
18.24 14.31 14.    7.25 38.07 23.95 25.71 17.31 29.93 10.65 12.43 24.08
11.69 13.42 14.26 15.95 12.48 29.8   8.52 14.52 11.38 22.82 19.08 20.27
11.17 12.26 18.26  8.51 10.33 14.15 16.   13.16 17.47 34.3  41.19 27.05
16.43  8.35 18.64 11.87  9.78  7.51 14.07 13.13 17.26 24.55 19.77 29.85
48.17 25.   13.39 16.49 21.5  12.66 16.21 13.81 17.51 24.52 20.76 31.71
10.59 10.63 50.81 15.81  7.25 31.85 16.82 32.9  17.89 14.48  9.6  34.63
34.65 23.33 45.35 23.17 40.55 20.69 20.9  30.46 18.15 23.1  15.69 19.81
28.44 15.48 16.58  7.56 10.34 43.11 13.   13.51 18.71 12.74 13.   16.4
20.53 16.47 26.59 38.73 24.27 12.76 30.06 25.89 48.33 13.27 28.17 12.9
28.15 11.59  7.74 30.14 12.16 13.42  8.58 15.98 13.42 16.27 10.09 20.45
13.28 22.12 24.01 15.69 11.61 10.77 15.53 10.07 12.6  32.83 35.83 29.03
27.18 22.67 17.82 18.78]
```



Viva Questions

1) **What is Machine learning?**
Machine learning is a branch of computer science which deals with system programming in order to automatically learn and improve with experience. For example: Robots are programed so that they can perform the task based on data they gather from sensors. It automatically learns programs from data.

2) **Mention the difference between Data Mining and Machine learning?**
Machine learning relates with the study, design and development of the algorithms that give computers the capability to learn without being explicitly programmed. While, data mining can be defined as the process in which the unstructured data tries to extract knowledge or unknown interesting patterns. During this process machine, learning algorithms are used.

3) **What is 'Overfitting' in Machine learning?**
In machine learning, when a statistical model describes random error or noise instead of underlying relationship 'overfitting' occurs. When a model is excessively complex, overfitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been overfit.

4) **Why overfitting happens?**
The possibility of overfitting exists as the criteria used for training the model is not the same as the criteria used to judge the efficacy of a model.

5) **How can you avoid overfitting ?**
By using a lot of data overfitting can be avoided, overfitting happens relatively as you have a small dataset, and you try to learn from it. But if you have a small database and you are forced to come with a model based on that. In such situation, you can use a technique known as **cross validation**. In this method the dataset splits into two section, testing and training datasets, the testing dataset will only test the model while, in training dataset, the datapoints will come up with the model.
In this technique, a model is usually given a dataset of a known data on which training (training data set) is run and a dataset of unknown data against which the model is tested. The idea of cross validation is to define a dataset to "test" the model in the training phase.

6) **What is inductive machine learning?**
The inductive machine learning involves the process of learning by examples, where a system, from a set of observed instances tries to induce a general rule.

7) **What are the five popular algorithms of Machine Learning?**
a) Decision Trees
b) Neural Networks (back propagation)
c) Probabilistic networks
d) Nearest Neighbor
e) Support vector machines

8) **What are the different Algorithm techniques in Machine Learning?**
The different types of techniques in Machine Learning are
a) Supervised Learning
b) Unsupervised Learning
c) Semi-supervised Learning
d) Reinforcement Learning
e) Transduction
f) Learning to Learn

9) **What are the three stages to build the hypotheses or model in machine learning?**
a) Model building
b) Model testing
c) Applying the model

10) **What is the standard approach to supervised learning?**
The standard approach to supervised learning is to split the set of example into the training set and the test.

11) **What is 'Training set' and 'Test set'?**
In various areas of information science like machine learning, a set of data is used to discover the potentially predictive relationship known as 'Training Set'. Training set is an examples given to the learner, while Test set is used to test the accuracy of the hypotheses generated by the learner, and it is the set of example held back from the learner. Training set are distinct from Test set.

12) **List down various approaches for machine learning?**
The different approaches in Machine Learning are
a) Concept Vs Classification Learning
b) Symbolic Vs Statistical Learning
c) Inductive Vs Analytical Learning

13) **What is not Machine Learning?**
a) Artificial Intelligence
b) Rule based inference

14) **Explain what is the function of 'Unsupervised Learning'?**
a) Find clusters of the data
b) Find low-dimensional representations of the data
c) Find interesting directions in data
d) Interesting coordinates and correlations
e) Find novel observations/ database cleaning

15) **Explain what is the function of 'Supervised Learning'?**
a) Classifications
b) Speech recognition
c) Regression
d) Predict time series
e) Annotate strings

16) What is algorithm independent machine learning?

Machine learning in where mathematical foundations is independent of any particular classifier or learning algorithm is referred as algorithm independent machine learning?

17) What is the difference between artificial learning and machine learning?

Designing and developing algorithms according to the behaviours based on empirical data are known as Machine Learning. While artificial intelligence in addition to machine learning, it also covers other aspects like knowledge representation, natural language processing, planning, robotics etc.

18) What is classifier in machine learning?

A classifier in a Machine Learning is a system that inputs a vector of discrete or continuous feature values and outputs a single discrete value, the class.

19) What are the advantages of Naive Bayes?

In Naïve Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. The main advantage is that it can’t learn interactions between features.

20) In what areas Pattern Recognition is used?

Pattern Recognition can be used in

a) Computer Vision
b) Speech Recognition
c) Data Mining
d) Statistics
e) Informal Retrieval
f) Bio-Informatics

21) What is Genetic Programming?

Genetic programming is one of the two techniques used in machine learning. The model is based on the testing and selecting the best choice among a set of results.

22) What is Inductive Logic Programming in Machine Learning?

Inductive Logic Programming (ILP) is a subfield of machine learning which uses logical programming representing background knowledge and examples.

23) What is Model Selection in Machine Learning?

The process of selecting models among different mathematical models, which are used to describe the same data set is known as Model Selection. Model selection is applied to the fields of statistics, machine learning and data mining.

24) What are the two methods used for the calibration in Supervised Learning?

The two methods used for predicting good probabilities in Supervised Learning are

a) Platt Calibration
b) Isotonic Regression

These methods are designed for binary classification, and it is not trivial.

25) Which method is frequently used to prevent overfitting?
When there is sufficient data 'Isotonic Regression' is used to prevent an overfitting issue.

26) What is the difference between heuristic for rule learning and heuristics for decision trees?
The difference is that the heuristics for decision trees evaluate the average quality of a number of disjointed sets while rule learners only evaluate the quality of the set of instances that is covered with the candidate rule.

27) What is Perceptron in Machine Learning?
In Machine Learning, Perceptron is an algorithm for supervised classification of the input into one of several possible non-binary outputs.

28) Explain the two components of Bayesian logic program?
Bayesian logic program consists of two components. The first component is a logical one ; it consists of a set of Bayesian Clauses, which captures the qualitative structure of the domain. The second component is a quantitative one, it encodes the quantitative information about the domain.

29) What are Bayesian Networks (BN) ?
Bayesian Network is used to represent the graphical model for probability relationship among a set of variables .

30) Why instance based learning algorithm sometimes referred as Lazy learning algorithm?
Instance based learning algorithm is also referred as Lazy learning algorithm as they delay the induction or generalization process until classification is performed.

31) What are the two classification methods that SVM (Support Vector Machine) can handle?
a) Combining binary classifiers
b) Modifying binary to incorporate multiclass learning

32) What is ensemble learning?
To solve a particular computational program, multiple models such as classifiers or experts are strategically generated and combined. This process is known as ensemble learning.

33) Why ensemble learning is used?
Ensemble learning is used to improve the classification, prediction, function approximation etc of a model.

34) When to use ensemble learning?
Ensemble learning is used when you build component classifiers that are more accurate and independent from each other.

35) What are the two paradigms of ensemble methods?
The two paradigms of ensemble methods are
a) Sequential ensemble methods
b) Parallel ensemble methods

36) **What is the general principle of an ensemble method and what is bagging and boosting in ensemble method?**
The general principle of an ensemble method is to combine the predictions of several models built with a given learning algorithm in order to improve robustness over a single model. Bagging is a method in ensemble for improving unstable estimation or classification schemes. While boosting method are used sequentially to reduce the bias of the combined model. Boosting and Bagging both can reduce errors by reducing the variance term.

37) **What is bias-variance decomposition of classification error in ensemble method?**
The expected error of a learning algorithm can be decomposed into bias and variance. A bias term measures how closely the average classifier produced by the learning algorithm matches the target function. The variance term measures how much the learning algorithm's prediction fluctuates for different training sets.

38) **What is an Incremental Learning algorithm in ensemble?**
Incremental learning method is the ability of an algorithm to learn from new data that may be available after classifier has already been generated from already available dataset.

39) **What is PCA, KPCA and ICA used for?**
PCA (Principal Components Analysis), KPCA (Kernel based Principal Component Analysis) and ICA (Independent Component Analysis) are important feature extraction techniques used for dimensionality reduction.

40) **What is dimension reduction in Machine Learning?**
In Machine Learning and statistics, dimension reduction is the process of reducing the number of random variables under considerations and can be divided into feature selection and feature extraction

41) **What are support vector machines?**
Support vector machines are supervised learning algorithms used for classification and regression analysis.

42) **What are the components of relational evaluation techniques?**
The important components of relational evaluation techniques are

a) Data Acquisition
b) Ground Truth Acquisition
c) Cross Validation Technique
d) Query Type
e) Scoring Metric
f) Significance Test

43) What are the different methods for Sequential Supervised Learning?

The different methods to solve Sequential Supervised Learning problems are

a) Sliding-window methods
b) Recurrent sliding windows
c) Hidden Markow models
d) Maximum entropy Markow models
e) Conditional random fields
f) Graph transformer networks

44) What are the areas in robotics and information processing where sequential prediction problem arises?

The areas in robotics and information processing where sequential prediction problem arises are

a) Imitation Learning
b) Structured prediction
c) Model based reinforcement learning

45) What is batch statistical learning?

Statistical learning techniques allow learning a function or predictor from a set of observed data that can make predictions about unseen or future data. These techniques provide guarantees on the performance of the learned predictor on the future unseen data based on a statistical assumption on the data generating process.

46) What is PAC Learning?

PAC (Probably Approximately Correct) learning is a learning framework that has been introduced to analyze learning algorithms and their statistical efficiency.

47) What are the different categories you can categorized the sequence learning process?

a) Sequence prediction
b) Sequence generation
c) Sequence recognition
d) Sequential decision

48) What is sequence learning?

Sequence learning is a method of teaching and learning in a logical manner.

49) What are two techniques of Machine Learning?

The two techniques of Machine Learning are

a) Genetic Programming
b) Inductive Learning

50) Give a popular application of machine learning that you see on day to day basis?

The recommendation engine implemented by major ecommerce websites uses Machine Lea