Dr.K.V.SUBBA REDDY INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

Approved By AICTE, New Delhi & Affiliated to JNTUA
Accredited by NAAC with A+ Grade, CSE Programme Accredited By NBA,
Recognized Under Section 2(f) And 12 B Of UGC Act 1956,
An ISO 9001:2000 Certified Institution
www.drkvsrit.ac.in

MACHINE LEARNING LAB (20A05602P) MANUAL

B.Tech (CSE) III Year/II Semester (R20)

Prepared By

Dr. Dhanaraj Cheelu

Professor, CSE Department, Dr.KVSRIT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr.K.V.SUBBA REDDY INSTITUTE OF TECHNOLOGY
Opp.Dupadu (RS) , N.H-44,Kurnool -518218, Kurnool District, A.P.

Dr.K.V.SUBBA REDDY INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)
Approved By AICTE, New Delhi & Affiliated to JNTUA
Accredited By NBA, Recognized Under Section 2(f) And 12 B Of UGC Act 1956,
An ISO 9001:2000 Certified Institution
www.drkvsrit.ac.in

Vision of Dr.KVSRIT:

To be a Global Leader in imparting Quality Technical Education to produce Competent, Technically Innovative Engineers imbibed with Research Aptitude, Entrepreneurship and Social Responsibility.

Mission of Dr.KVSRIT:

1. To Nurture the Students with Fundamental Engineering Knowledge enriched with Technical Skills.
2. To Create Conducive Environment to nurture Innovation and Interdisciplinary Research.
3. To Develop Professionals through Innovative Pedagogy focusing on Individual Growth, Discipline, Integrity, Ethics and Social Responsibility.
4. To Foster Industry-Institution Partnerships Leading to Skill Development and Entrepreneurship.

Vision of Computer Science and Engineering Department:

To be a Center for Academic Excellence in Computer Science and Engineering Education, Research and Consultancy Contributing Effectively to meet Industrial and Societal needs.

Mission of Computer Science and Engineering Department:

M1: To Impart quality technical education with global standards.
M2: To Provide a platform for harnessing Industry oriented technical skills with inter - disciplinary research awareness.
M3: To Promote entrepreneurship and leadership qualities imbibed with professional ethics.

Program Outcomes (PO):

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 analyse 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.

Programme Specific Outcomes (PSOs):

PSO1: Apply Software Engineering Principles and Practices to provide software solutions.

PSO2: Design and Develop Network and Mobile based applications.

PSO3: Design innovative algorithms and develop effective code for business applications.

Program Educational Objectives (PEO):

Graduates of the Program will have

PEO1: Strong fundamental knowledge in Computer Science & Engineering, technical competency and problem-solving skills to develop innovative solutions.

PEO2: Necessary domain knowledge and successful professional career in IT and allied fields of Computer Science & Engineering.

PEO3: Ability to pursue higher education and Entrepreneurship.

PEO4: Necessary skills for lifelong learning, teamwork and research to cater for real time needs of industry and society.

MACHINE LEARNING LAB (20A05602P)

Common to CSE, CSD, CSE(AI),CSE(AI&ML),CSE(DS),AI&DS

Course Objectives:

1. Make use of Data sets in implementing the machine learning algorithms
2. Implement the machine learning concepts and algorithms in any suitable language of choice.

Course Outcomes (CO):

After completion of the course, students will be able to

1. Understand the Mathematical and statistical prospective of machine learning algorithms
2. through python programming
3. Appreciate the importance of visualization in the data analytics solution.
4. Derive insights using Machine learning algorithms

Software & Hardware Requirements

The following Software can be used for Python programming:

Online Compilers:

1. **Vim**
   Vim is a highly configurable and special text editor used by programmers. It is a powerful tool that enhances editing files, performs search and replace operations, and much more. Vim also supports syntax highlighting, which helps to make code more readable. Vim is especially popular among experienced coders and is often considered the most powerful text editor available.
2. **GNU Emacs**
   GNU Emacs is a free and open-source text editor. Richard Stallman created it in 1985. Emacs is a powerful and extensible text editor. It has a rich set of features, including a built-in Lisp interpreter, powerful editing features, and support for a wide range of programming languages and file formats.
3. **ActivePython**
   ActivePython is a distribution of Python created by ActiveState, which is available for Windows, Linux, and macOS X. ActivePython also includes many additional packages not found in the standard Python distribution. These include packages for scientific computing, data analysis, and web development. ActivePython is free to download and use for development purposes.

Python IDEs:

1. Anaconda Spyder IDE
2. IDLE
3. PyCharm
4. Visual Studio Code
5. Sublime Text 3
6. Atom
7. Jupyter
8. Spyder
9. PyDev
10. Thonny
11. Wing

Hardware Requirements:

Preferable Modern Operating System requirements (Not Mandatory):

i. Windows 7 or 10
ii. Mac OS X 10.11 or higher, 64-bit
iii. Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
iv. X86 64-bit CPU (Intel / AMD architecture). ARM CPUs are not supported.
v. 4 GB RAM
vi. 5 GB free disk space

General Guidelines for Learning Python Programming Effectively

1. **Make It Stick**

    Tip #1: Code Everyday

    Tip #2: Write It Out

    Tip #3: Go Interactive!

    Tip #4: Take Breaks

    Tip #5: Become a Bug Bounty Hunter

2. **Make It Collaborative**

    Tip #6: Surround Yourself With Others Who Are Learning

    Tip #7: Teach

    Tip #8: Pair Program

    Tip #9: Ask “GOOD” Questions

3. **Make Something**

    Tip #10: Build Something, Anything

    Tip #11: Contribute to Open Source

MACHINE LEARNING LAB (20A05602P)

INDEX OF EXPERIEMENTS

List of Experiments:

Note:

a. The programs can be implemented in either JAVA or Python.
b. For Problems 1 to 6 and 10, programs are to be developed without using the built-in classes or APIs of Java/Python.
c. Data sets can be taken from standard repositories (https://archive.ics.uci.edu/ml/datasets.html) or constructed by the students.

Projects:

1. Predicting the Sale price of a house using Linear regression
2. Spam classification using Naïve Bayes algorithm
3. Predict car sale prices using Artificial Neural Networks
4. Predict Stock market trends using LSTM
5. Detecting faces from images

References:

1) Python Machine Learning Workbook for beginners, AI Publishing, 2020.

Online Learning Resources/Virtual Labs:

1) Machine Learning A-Z (Python & R in Data Science Course) | Udemy
2) Machine Learning | Coursera

Exp. No. 1. Implement and demonstrate the FIND-S algorithm in Python for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file

Find-S Algorithm Machine Learning

1. Initilize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute contraint a_i in h
      If the contraint a_i is satisfied by x
         then do nothing
      Else
         replace a_i in h by the next more general constraint that is satisfied by x
3. Output the hypothesis h

Python Program to Implement and Demonstrate FIND-S Algorithm

```
"""
1. Implement and demonstrate the FIND-S algorithm for finding the
most specific hypothesis based on a given set of training data samples.
Read the training data from a .CSV file
"""
import csv
a = []
with open('enjoysport.csv', 'r') as csvfile:
   next(csvfile)
   for row in csv.reader(csvfile):
      a.append(row)
   print(a)

print("\nThe total number of training instances are : ",len(a))

num_attribute = len(a[0])-1

print("\nThe initial hypothesis is : ")
hypothesis = ['0']*num_attribute
print(hypothesis)

for i in range(0, len(a)):
   if a[i][num_attribute] == 'yes':
      print ("\nInstance ", i+1, "is", a[i], " and is Positive Instance")
      for j in range(0, num_attribute):
         if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
            hypothesis[j] = a[i][j]
         else:
            hypothesis[j] = '?'
      print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")

   if a[i][num_attribute] == 'no':
      print ("\nInstance ", i+1, "is", a[i], " and is Negative Instance Hence Ignored")
      print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")

print("\nThe Maximally specific hypothesis for the training instance is ", hypothesis)
```

Output:

EnjoySport Dataset is saved as .csv (comma separated values) file in the current working directory otherwise use the complete path of the dataset set in the program:

sky	airtemp	humidity	wind	water	forcast	enjoysport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

[[‘sky’, ‘airtemp’, ‘humidity’, ‘wind’, ‘water’, ‘forcast’, ‘enjoysport’], [‘sunny’, ‘warm’, ‘normal’, ‘strong’, ‘warm’, ‘same’, ‘yes’], [‘sunny’, ‘warm’, ‘high’, ‘strong’, ‘warm’, ‘same’, ‘yes’], [‘rainy’, ‘cold’, ‘high’, ‘strong’, ‘warm’, ‘change’, ‘no’], [‘sunny’, ‘warm’, ‘high’, ‘strong’, ‘cool’, ‘change’, ‘yes’]]

The total number of training instances are : 5
The initial hypothesis is : [‘0’, ‘0’, ‘0’, ‘0’, ‘0’, ‘0’]
Instance 2 is [‘sunny’, ‘warm’, ‘normal’, ‘strong’, ‘warm’, ‘same’, ‘yes’] and is Positive Instance
The hypothesis for the training instance 2 is: [‘sunny’, ‘warm’, ‘normal’, ‘strong’, ‘warm’, ‘same’]
Instance 3 is [‘sunny’, ‘warm’, ‘high’, ‘strong’, ‘warm’, ‘same’, ‘yes’] and is Positive Instance
The hypothesis for the training instance 3 is: [‘sunny’, ‘warm’, ‘?’, ‘strong’, ‘warm’, ‘same’]
Instance 4 is [‘rainy’, ‘cold’, ‘high’, ‘strong’, ‘warm’, ‘change’, ‘no’] and is Negative Instance Hence Ignored
The hypothesis for the training instance 4 is: [‘sunny’, ‘warm’, ‘?’, ‘strong’, ‘warm’, ‘same’]
Instance 5 is [‘sunny’, ‘warm’, ‘high’, ‘strong’, ‘cool’, ‘change’, ‘yes’] and is Positive Instance
The hypothesis for the training instance 5 is: [‘sunny’, ‘warm’, ‘?’, ‘strong’, ‘?’, ‘?’]
The Maximally specific hypothesis for the training instance is [‘sunny’, ‘warm’, ‘?’, ‘strong’, ‘?’, ‘?’]

Exp. No. 2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm in python to output a description of the set of all hypotheses consistent with the training examples.

Candidate Elimination Algorithm Machine Learning

```
For each training example d, do:
    If d is positive example
        Remove from G any hypothesis h inconsistent with d
        For each hypothesis s in S not consistent with d:
            Remove s from S
            Add to S all minimal generalizations of s consistent with d and having a generalization in G
            Remove from S any hypothesis with a more specific h in S
    If d is negative example
        Remove from S any hypothesis h inconsistent with d
        For each hypothesis g in G not consistent with d:
            Remove g from G
            Add to G all minimal specializations of g consistent with d and having a specialization in S
            Remove from G any hypothesis having a more general hypothesis in G
```

Python Program to Implement and Demonstrate FIND-S Algorithm

```
import numpy as np
import pandas as pd

data = pd.read_csv(path+'/enjoysport.csv')
concepts = np.array(data.iloc[:,0:-1])
print("\nInstances are:\n",concepts)
target = np.array(data.iloc[:,-1])
print("\nTarget Values are: ",target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and genearal_h")
    print("\nSpecific Boundary: ", specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)

    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "yes":
            print("Instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("Instance is Negative ")
            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("Specific Bundary after ", i+1, "Instance is ", specific_h)
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h

s_final, g_final = learn(concepts, target)
```

```
print("Final Specific_h: ", s_final, sep="\n")
print("Final General_h: ", g_final, sep="\n")
```

Dataset:

EnjoySport Dataset is saved as .csv (comma separated values) file in the current working directory otherwise use the complete path of the dataset set in the program:

Output:

Instances are:
[[‘sunny’ ‘warm’ ‘normal’ ‘strong’ ‘warm’ ‘same’]
[‘sunny’ ‘warm’ ‘high’ ‘strong’ ‘warm’ ‘same’]
[‘rainy’ ‘cold’ ‘high’ ‘strong’ ‘warm’ ‘change’]
[‘sunny’ ‘warm’ ‘high’ ‘strong’ ‘cool’ ‘change’]]
Target Values are: [‘yes’ ‘yes’ ‘no’ ‘yes’]
Initialization of specific_h and genearal_h
Specific Boundary: [‘sunny’ ‘warm’ ‘normal’ ‘strong’ ‘warm’ ‘same’]
Generic Boundary: [[‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’]]
Instance 1 is [‘sunny’ ‘warm’ ‘normal’ ‘strong’ ‘warm’ ‘same’] Instance is Positive
Specific Bundary after 1 Instance is [‘sunny’ ‘warm’ ‘normal’ ‘strong’ ‘warm’ ‘same’]
Generic Boundary after 1 Instance is [[‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’]]
Instance 2 is [‘sunny’ ‘warm’ ‘high’ ‘strong’ ‘warm’ ‘same’] Instance is Positive
Specific Bundary after 2 Instance is [‘sunny’ ‘warm’ ‘?’ ‘strong’ ‘warm’ ‘same’]
Generic Boundary after 2 Instance is [[‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’]]
Instance 3 is [‘rainy’ ‘cold’ ‘high’ ‘strong’ ‘warm’ ‘change’] Instance is Negative
Specific Bundary after 3 Instance is [‘sunny’ ‘warm’ ‘?’ ‘strong’ ‘warm’ ‘same’]
Generic Boundary after 3 Instance is [[‘sunny’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘warm’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘same’]]
Instance 4 is [‘sunny’ ‘warm’ ‘high’ ‘strong’ ‘cool’ ‘change’] Instance is Positive
Specific Bundary after 4 Instance is [‘sunny’ ‘warm’ ‘?’ ‘strong’ ‘?’ ‘?’]
Generic Boundary after 4 Instance is [[‘sunny’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘warm’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’]]
Final Specific_h: [‘sunny’ ‘warm’ ‘?’ ‘strong’ ‘?’ ‘?’]
Final General_h: [[‘sunny’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’], [‘?’, ‘warm’, ‘?’, ‘?’, ‘?’, ‘?’]]

Exp. No. 3. 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.

Decision Tree ID3 Algorithm Machine Learning

```
ID3(Examples, Target_attribute, Attributes)

Examples are the training examples.
Target_attribute is the attribute whose value is to be predicted by the tree.
Attributes is a list of other attributes that may be tested by the learned decision tree.
Returns a decision tree that correctly classifies the given Examples.

Create a Root node for the tree
If all Examples are positive, Return the single-node tree Root, with label = +
If all Examples are negative, Return the single-node tree Root, with label = -
If Attributes is empty, Return the single-node tree Root,
with label = most common value of Target_attribute in Examples

Otherwise Begin
  A ← the attribute from Attributes that best* classifies Examples
  The decision attribute for Root ← A
  For each possible value, vi, of A,
    Add a new tree branch below Root, corresponding to the test A = vi
    Let Examples vi, be the subset of Examples that have value vi for A
    If Examples vi , is empty
      Then below this new branch add a leaf node with
      label = most common value of Target_attribute in Examples
    Else
      below this new branch add the subtree
      ID3(Examples vi, Targe_tattribute, Attributes – {A}))
End
Return Root
```

The best attribute is the one with highest information gain

ENTROPY:
Entropy measures the impurity of a collection of examples.

$$Entropy\ (S) \equiv -p_{\oplus} log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus}$$

Where, p_+ is the proportion of positive examples in S
p_- is the proportion of negative examples in S.

INFORMATION GAIN:
Information gain, is the expected reduction in entropy caused by partitioning the examples according to this attribute.
The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Dataset:

PlayTennis Dataset is saved as .csv (comma separated values) file in the current working directory otherwise use the complete path of the dataset set in the program:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Python Program to Implement and Demonstrate FIND-S Algorithm

Import libraries and read data using read_csv() function. Remove the target from the data and store attributes in the features variable.

```
import pandas as pd
import math
import numpy as np

data = pd.read_csv("Dataset/4-dataset.csv")
features = [feat for feat in data]
features.remove("answer")
```

Create a class named Node with four members children, value, isLeaf and pred.

```
class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""
```

Define a function called entropy to find the entropy oof the dataset.

```
def entropy(examples):
    pos = 0.0
    neg = 0.0
    for _, row in examples.iterrows():
        if row["answer"] == "yes":
            pos += 1
        else:
            neg += 1
    if pos == 0.0 or neg == 0.0:
        return 0.0
    else:
        p = pos / (pos + neg)
        n = neg / (pos + neg)
        return -(p * math.log(p, 2) + n * math.log(n, 2))
```

Define a function named info_gain to find the gain of the attribute

```
def info_gain(examples, attr):
    uniq = np.unique(examples[attr])
    #print ("\n",uniq)
    gain = entropy(examples)
    #print ("\n",gain)
    for u in uniq:
        subdata = examples[examples[attr] == u]
        #print ("\n",subdata)
        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
        #print ("\n",gain)
    return gain
```

Define a function named ID3 to get the decision tree for the given dataset

```
def ID3(examples, attrs):
    root = Node()

    max_gain = 0
    max_feat = ""
    for feature in attrs:
        #print ("\n",examples)
        gain = info_gain(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat
    #print ("\nMax feature attr",max_feat)
    uniq = np.unique(examples[max_feat])
    #print ("\n",uniq)
    for u in uniq:
        #print ("\n",u)
        subdata = examples[examples[max_feat] == u]
        #print ("\n",subdata)
        if entropy(subdata) == 0.0:
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["answer"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)

    return root
```

Define a function named printTree to draw the decision tree

```
def printTree(root: Node, depth=0):
    for i in range(depth):
        print("\t", end="")
    print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    print()
    for child in root.children:
        printTree(child, depth + 1)
```

Define a function named classify to classify the new example

```
def classify(root: Node, new):
    for child in root.children:
        if child.value == new[root.value]:
            if child.isLeaf:
                print ("Predicted Label for new example", new," is:", child.pred)
                exit
            else:
                classify (child.children[0], new)
```

Finally, call the ID3, printTree and classify functions

```
root = ID3(data, features)
print("Decision Tree is:")
printTree(root)
print ("------------------")

new = {"outlook":"sunny", "temperature":"hot", "humidity":"normal", "wind":"strong"}
classify (root, new)
```

Output:


```
Decision Tree is:
outlook
        overcast ->  ['yes']
        rain
                wind
                        strong ->  ['no']
                        weak ->  ['yes']
        sunny
                humidity
                        high ->  ['no']
                        normal ->  ['yes']

------------------
Predicted Label for new example {'outlook': 'sunny', 'temperature': 'hot', 'humidity': 'normal', 'wind': 'strong'}  is: ['yes']
```

Exp. No. 4. Build an Artificial Neural Network by implementing the Back-propagation 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) #maximum of X array longitudinally
y = y/100


#Sigmoid Function
def sigmoid (x):
    return 1/(1 + np.exp(-x))


#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)


#Variable initialization
epoch=5 #Setting training iterations
lr=0.1 #Setting learning rate


inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization


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))

#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):
    #Forward Propogation
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp= outinp1+bout
    output = sigmoid(outinp)

    #Backpropagation
    EO = y-output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)#how much hidden layer wts contributed to error
    d_hiddenlayer = EH * hiddengrad

    wout += hlayer_act.T.dot(d_output) *lr   # dotproduct of nextlayererror and currentlayerop
    wh += X.T.dot(d_hiddenlayer) *lr

    print ("-----------Epoch-", i+1, "Starts----------")
    print("Input: \n" + str(X))
    print("Actual Output: \n" + str(y))
```

```
    print("Predicted Output: \n" ,output)
    print ("-----------Epoch-", i+1, "Ends----------\n")

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

Training Examples:

Example	Sleep	Study	Expected % in Exams
1	2	9	92
2	1	5	86
3	3	6	89

Normalize the input

Example	Sleep	Study	Expected % in Exams
1	2/3 = 0.66666667	9/9 = 1	0.92
2	1/3 = 0.33333333	5/9 = 0.55555556	0.86
3	3/3 = 1	6/9 = 0.66666667	0.89

Output

————Epoch- 1 Starts————-
Input:
[[0.66666667 1.]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]

[0.89]]
Predicted Output:
[[0.81951208]
[0.8007242]
[0.82485744]]
————Epoch- 1 Ends————-

————Epoch- 2 Starts————-
Input:
[[0.66666667 1.]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]
[0.89]]
Predicted Output:
[[0.82033938]
[0.80153634]
[0.82568134]]
————Epoch- 2 Ends————-

————Epoch- 3 Starts————-
Input:
[[0.66666667 1.]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]
[0.89]]
Predicted Output:
[[0.82115226]
[0.80233463]
[0.82649072]]
————Epoch- 3 Ends————-

————Epoch- 4 Starts————-
Input:
[[0.66666667 1.]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]
[0.89]]
Predicted Output:
[[0.82195108]
[0.80311943]
[0.82728598]]
————Epoch- 4 Ends————-

Exp. No. 5. 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.

Bayes' Theorem is stated as:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Where,

P(h|D) is the probability of hypothesis h given the data D. This is called the **posterior probability**.
P(D|h) is the probability of data d given that the hypothesis h was true.
P(h) is the probability of hypothesis h being true. This is called the **prior probability of h. P(D)** is the probability of the data. This is called the **prior probability of D**

After calculating the posterior probability for a number of different hypotheses h, and is interested in finding the most probable hypothesis h ∈ H given the observed data D. Any such maximally probable hypothesis is called a ***maximum a posteriori (MAP) hypothesis***.
Bayes theorem to calculate the posterior probability of each candidate hypothesis is ***hMAP*** is a MAP hypothesis provided.

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

(Ignoring P(D) since it is a constant)

Gaussian Naive Bayes

A Gaussian Naive Bayes algorithm is a special type of Naïve Bayes algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a Gaussian distribution i.e., normal distribution

Representation for Gaussian Naive Bayes

We calculate the probabilities for input values for each class using a frequency. With real- valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution.

This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

Gaussian Naive Bayes Model from Data

The probability density function for the normal distribution is defined by two parameters (mean and standard deviation) and calculating the mean and standard deviation values of each input variable (x) for each class value.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$ Mean

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2\right]^{0.5}$$ Standard deviation

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ Normal distribution

Examples:

The data set used in this program is the ***Pima Indians Diabetes problem***.

This data set is comprised of 768 observations of medical details for Pima Indians patents. The records describe instantaneous measurements taken from the patient such as their age, the number of times pregnant and blood workup. All patients are women aged 21 or older. All attributes are numeric, and their units vary from attribute to attribute.

The attributes are Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabeticPedigreeFunction, Age, Outcome

Each record has a class value that indicates whether the patient suffered an onset of diabetes within 5 years of when the measurements were taken (1) or not (0)

Sample Examples:

Examples	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetic Pedigree Function	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	1
10	8	125	96	0	0	0	0.232	54	1

Python Program to Implement and Demonstrate Naïve Bayesian Classifier Machine Learning

```
import csv
import random
import math

def loadcsv(filename):
	lines = csv.reader(open(filename, "r"));
	dataset = list(lines)
	for i in range(len(dataset)):
    #converting strings into numbers for processing
		dataset[i] = [float(x) for x in dataset[i]]

	return dataset

def splitdataset(dataset, splitratio):
  #67% training size
	trainsize = int(len(dataset) * splitratio);
	trainset = []
	copy = list(dataset);
	while len(trainset) < trainsize:
#generate indices for the dataset list randomly to pick ele for training data
		index = random.randrange(len(copy));
		trainset.append(copy.pop(index))
	return [trainset, copy]

def separatebyclass(dataset):
	separated = {} #dictionary of classes 1 and 0
#creates a dictionary of classes 1 and 0 where the values are
#the instances belonging to each class
	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): #creates a dictionary of classes
	summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)];
	del summaries[-1] #excluding labels +ve or -ve
	return summaries
```

```
def summarizebyclass(dataset):
        separated = separatebyclass(dataset);
   #print(separated)
        summaries = {}
        for classvalue, instances in separated.items():
#for key,value in dic.items()
#summaries is a dic of tuples(mean,std) for each class value
                summaries[classvalue] = summarize(instances) #summarize is used to cal to mean and std
        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 = {} # probabilities contains the all prob of all class of test data
        for classvalue, classsummaries in summaries.items():#class and attribute information as mean and sd
                probabilities[classvalue] = 1
                for i in range(len(classsummaries)):
                        mean, stdev = classsummaries[i] #take mean and sd of every attribute for class 0 and 1 seperaely
                        x = inputvector[i] #testvector's first attribute
                        probabilities[classvalue] *= calculateprobability(x, mean, stdev);#use normal dist
        return probabilities

def predict(summaries, inputvector): #training and test data is passed
        probabilities = calculateclassprobabilities(summaries, inputvector)
        bestLabel, bestProb = None, -1
        for classvalue, probability in probabilities.items():#assigns that class which has he highest prob
                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 (correct/float(len(testset))) * 100.0

def main():
        filename = 'naivedata.csv'
        splitratio = 0.67
        dataset = loadcsv(filename);

        trainingset, testset = splitdataset(dataset, splitratio)
        print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingset), len(testset)))
```

```
            # prepare model
            summaries = summarizebyclass(trainingset);
            #print(summaries)
  # test model
            predictions = getpredictions(summaries, testset) #find the predictions of test data with the training data
            accuracy = getaccuracy(testset, predictions)
            print('Accuracy of the classifier is : {0}%'.format(accuracy))

main()
```

Output

Split 768 rows into train=514 and test=254

Rows Accuracy of the classifier is : 71.65354330708661%

Exp. No. 6. Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

Bayes' Theorem is stated as:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Where,

P(h|D) is the probability of hypothesis h given the data D. This is called the **posterior probability**.
P(D|h) is the probability of data d given that the hypothesis h was true.

P(h) is the probability of hypothesis h being true. This is called the **prior probability of h. P(D)** is the probability of the data. This is called the **prior probability of D**

After calculating the posterior probability for a number of different hypotheses h, and is interested in finding the most probable hypothesis h ∈ H given the observed data D. Any such maximally probable hypothesis is called a ***maximum a posteriori (MAP) hypothesis***.
Bayes theorem to calculate the posterior probability of each candidate hypothesis is ***hMAP*** is a MAP hypothesis provided.

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

(Ignoring P(D) since it is a constant)

CLASSIFY_NAIVE_BAYES_TEXT (Doc)

Return the estimated target value for the document Doc. ai denotes the word found in the ith position within Doc.

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return *VNB,* where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Data set:

	Text Documents	**Label**
1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg

Python Program to Implement and Demonstrate Naïve Bayesian Classifier using API for document classification

```
"""
6. Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task.
Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set

"

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

msg=pd.read_csv('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

#splitting the dataset into train and test data
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 words or Tokens in the text documents
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())

# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
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

```
The dimensions of the dataset (18, 2)
1. I love this sandwich
2. This is an amazing place
3. I feel very good about these beers
4. This is my best work
5. What an awesome view
6. I do not like this restaurant
7. I am tired of this stuff
8. I can’t deal with this
```

9. He is my sworn enemy
10. My boss is horrible
11. This is an awesome place
12. I do not like the taste of this juice
13. I love to dance
14. I am sick and tired of this place
15. What a great holiday
16. That is a bad locality to stay
17. We will have good fun tomorrow
18. 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
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
The total number of Training Data: (13,) The total number of Test Data: (5,)

The words or Tokens in the text documents
['about', 'am', 'amazing', 'an', 'and', 'awesome', 'beers', 'best', 'can', 'deal', 'do', 'enemy', 'feel',
'fun', 'good', 'great', 'have', 'he', 'holiday', 'house', 'is', 'like', 'love', 'my', 'not', 'of', 'place',
'restaurant', 'sandwich', 'sick', 'sworn', 'these', 'this', 'tired', 'to', 'today', 'tomorrow', 'very', 'view', 'we', 'went', 'what', 'will',
'with', 'work']
Accuracy of the classifier is 0.8
Confusion matrix
[[2 1]

[0 2]]

The value of Precision 0.6666666666666666
The value of Recall 1.0

Exp. No. 7. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using a standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

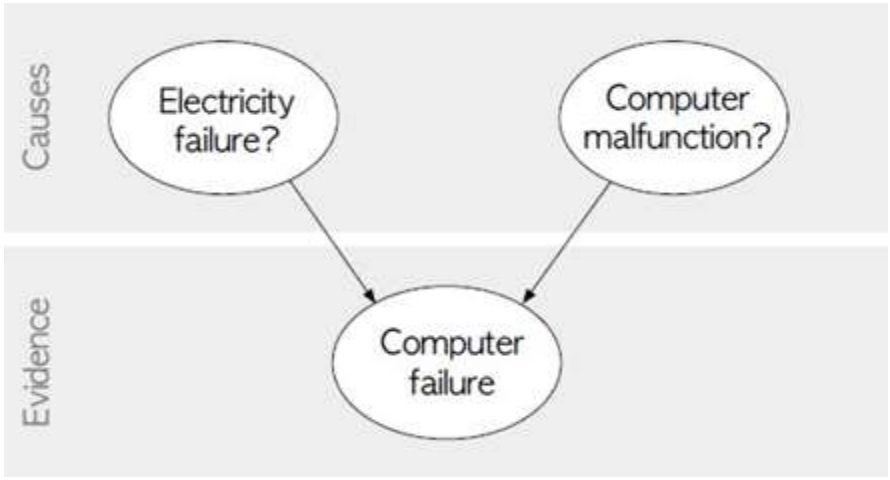
A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable.

Bayesian network consists of two major parts: a directed acyclic graph and a set of conditional probability distributions

- The directed acyclic graph is a set of random variables represented by nodes.
- The conditional probability distribution of a node (random variable) is defined for every possible outcome of the preceding causal node(s).

For illustration, consider the following example. Suppose we attempt to turn on our computer, but the computer does not start (observation/evidence). We would like to know which of the possible causes of computer failure is more likely. In this simplified illustration, we assume only two possible causes of this misfortune: electricity failure and computer malfunction.

The corresponding directed acyclic graph is depicted in below figure.



The goal is to calculate the posterior conditional probability distribution of each of the possible unobserved causes given the observed evidence, i.e. P [Cause | Evidence].

Data Set:

Title: Heart Disease Databases
The Cleveland database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "Heartdisease" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

Database: 0 1 2 3 4 Total

Cleveland: 164 55 36 35 13 303

Attribute Information:

1. age: age in years

2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
    1. Value 1: typical angina
    2. Value 2: atypical angina
    3. Value 3: non-anginal pain
    4. Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholestoral in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
    1. Value 0: normal
    2. Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    3. Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    1. Value 1: upsloping
    2. Value 2: flat
    3. Value 3: downsloping
12. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
13. Heartdisease: It is integer valued from 0 (no presence) to 4.

Some instance from the dataset:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heartdisease
63	1	1	145	233	1	2	150	o	2.3	3	o	6	o
67	1	4	160	286	o	2	108	1	1.5	2	3	3	2
67	1	4	120	229	o	2	129	1	2.6	2	2	7	1
41	o	2	130	204	o	2	172	o	1.4	1	o	3	o
62	o	4	140	268	o	2	160	o	3.6	3	2	3	3
60	1	4	130	206	o	2	132	1	2.4	2	2	7	4

Python Program to Implement and Demonstrate Bayesian network using pgmpy Machine Learning

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)

print('Sample instances from the dataset are given below')
print(heartDisease.head())

print('\n Attributes and datatypes')
print(heartDisease.dtypes)
```

```
model=
BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','chol')])
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)

print('\n 1. Probability of HeartDisease given evidence= restecg')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)

print('\n 2. Probability of HeartDisease given evidence= cp ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

Output

```
Learning CPD using Maximum likelihood estimators

 Inferencing with Bayesian Network:

 1. Probability of HeartDisease given evidence= restecg
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.1012 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.2392 |
+-----------------+---------------------+
| heartdisease(3) |              0.2015 |
+-----------------+---------------------+
| heartdisease(4) |              0.4581 |
+-----------------+---------------------+
2. Probability of HeartDisease given evidence= cp
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

Exp. No. 8. Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using the 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.

Python Program to Implement and Demonstrate K-Means and EM Algorithm Machine Learning

```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import sklearn.metrics as metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']

dataset = pd.read_csv("8-dataset.csv", names=names)

X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])

# K-PLOT
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
plt.title('KMeans')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])

print('The accuracy score of K-Mean: ',metrics.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y, model.labels_))

# GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y, y_cluster_gmm))
```

Output

The accuracy score of K-Mean: 0.24

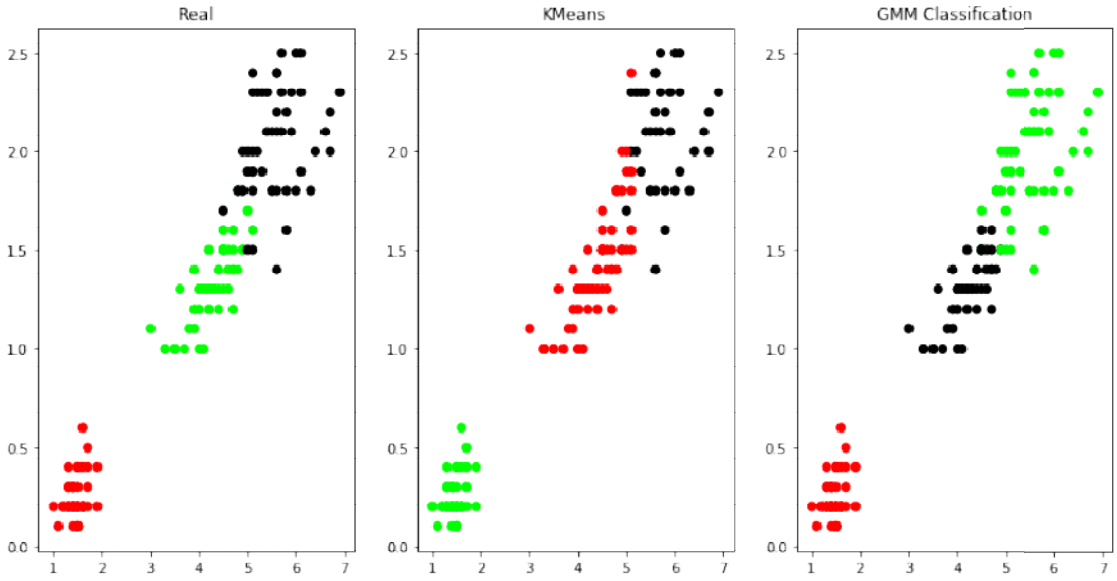
The Confusion matrixof K-Mean:

[[0 50 0]
[48 0 2]
[14 0 36]]

The accuracy score of EM: 0.36666666666666664

The Confusion matrix of EM:

[[50 0 0]
[0 5 45]
[0 50 0]]



Exp. No. 9. 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.

K-Nearest Neighbor Algorithm:

Training algorithm:

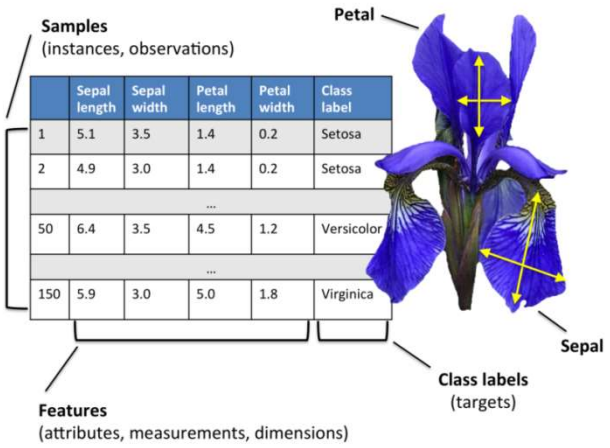
- For each training example (x, f (x)), add the example to the list training examples Classification algorithm:
  - Given a query instance x_q to be classified,
    - Let $x_1 \ldots x_k$ denote the k instances from training examples that are nearest to x_q
    - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

- Where, $f(x_i)$ function to calculate the mean value of the k nearest training examples.

Data Set:

Iris Plants Dataset: Dataset contains 150 instances (50 in each of three classes) Number of Attributes: 4 numeric, predictive attributes and the Class.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Sample Data

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Python Program to Implement and Demonstrate KNN Algorithm

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
dataset = pd.read_csv("9-dataset.csv", names=names)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
print(X.head())
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.10)

classifier = KNeighborsClassifier(n_neighbors=5).fit(Xtrain, ytrain)

ypred = classifier.predict(Xtest)

i = 0
print ("\n-------------------------------------------------------------------------")
print ('%-25s %-25s %-25s' % ('Original Label', 'Predicted Label', 'Correct/Wrong'))
print ("-------------------------------------------------------------------------")
for label in ytest:
    print ('%-25s %-25s' % (label, ypred[i]), end="")
    if (label == ypred[i]):
        print (' %-25s' % ('Correct'))
    else:
        print (' %-25s' % ('Wrong'))
    i = i + 1
print ("-------------------------------------------------------------------------")
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print ("-------------------------------------------------------------------------")
print("\nClassification Report:\n",metrics.classification_report(ytest, ypred))
print ("-------------------------------------------------------------------------")
print('Accuracy of the classifer is %0.2f' % metrics.accuracy_score(ytest,ypred))
print ("-------------------------------------------------------------------------")
```

Output

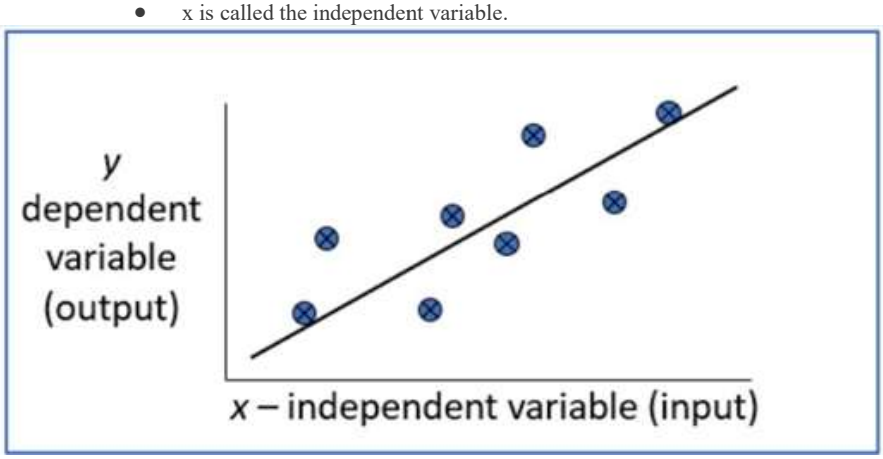
```
   sepal-length  sepal-width  petal-length  petal-width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2

-------------------------------------------------------------------------
Original Label          Predicted Label          Correct/Wrong
-------------------------------------------------------------------------
Iris-versicolor          Iris-versicolor          Correct
Iris-virginica           Iris-versicolor          Wrong
Iris-virginica           Iris-virginica           Correct
Iris-versicolor          Iris-versicolor          Correct
Iris-setosa              Iris-setosa              Correct
Iris-versicolor          Iris-versicolor          Correct
Iris-setosa              Iris-setosa              Correct
Iris-setosa              Iris-setosa              Correct
Iris-virginica           Iris-virginica           Correct
Iris-virginica           Iris-versicolor          Wrong
Iris-virginica           Iris-virginica           Correct
Iris-setosa              Iris-setosa              Correct
Iris-virginica           Iris-virginica           Correct
Iris-virginica           Iris-virginica           Correct
Iris-versicolor          Iris-versicolor          Correct
-------------------------------------------------------------------------

Confusion Matrix:
 [[4 0 0]
 [0 4 0]
 [0 2 5]]
-------------------------------------------------------------------------

Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         4
Iris-versicolor       0.67      1.00      0.80         4
 Iris-virginica       1.00      0.71      0.83         7

    avg / total       0.91      0.87      0.87        15

-------------------------------------------------------------------------
Accuracy of the classifer is 0.87
-------------------------------------------------------------------------
```

Exp. No. 10. Implement the non-parametric Locally Weighted Regression algorithm in Python in order to fit data points. Select the appropriate data set for your experiment and draw graphs.

Locally Weighted Regression Algorithm:

Regression:

- Regression is a technique from statistics that are used to predict values of the desired target quantity when the target quantity is continuous.
  - In regression, we seek to identify (or estimate) a continuous variable y associated with a given input vector x.
    - y is called the dependent variable.
    - x is called the independent variable.



Loess/Lowess Regression:

Loess regression is a nonparametric technique that uses local weighted regression to fit a smooth curve through points in a scatter plot.

Lowess Algorithm:

Locally weighted regression is a very powerful nonparametric model used in statistical learning.
Given a dataset X, y, we attempt to find a model parameter β(x) that minimizes residual sum of weighted squared errors.
The weights are given by a kernel function (k or w) which can be chosen arbitrarily

Algorithm

1. Read the Given data Sample to X and the curve (linear or non linear) to Y
2. Set the value for Smoothening parameter or Free parameter say τ
3. Set the bias /Point of interest set x0 which is a subset of X
4. Determine the weight matrix using :

$$w(x, x_o) = e^{-\frac{(x-x_o)^2}{2\tau^2}}$$

5. Determine the value of model term parameter β using:

$$\hat{\beta}(x_o) = (X^T W X)^{-1} X^T W y$$

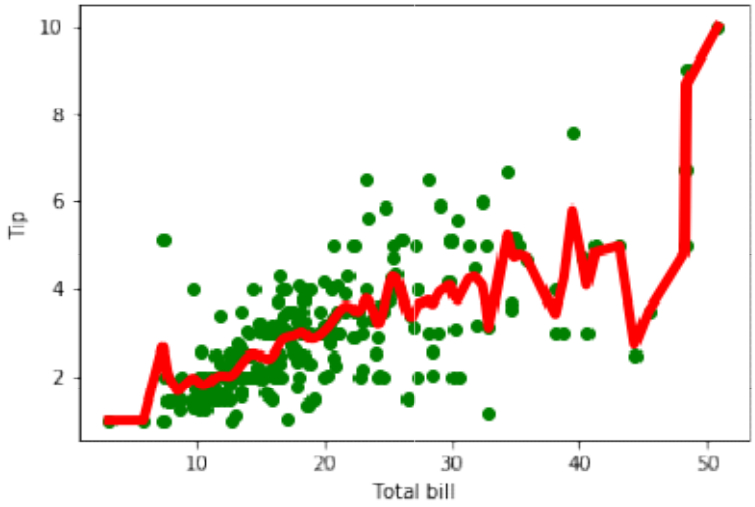
6. Prediction = x0*β

Python Program to Implement and Demonstrate Locally Weighted Regression Algorithm

```
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(np1.eye((m)))
    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

# load data points
data = pd.read_csv('10-dataset.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)

#preparing and add 1 in bill
mbill = np.mat(bill)
mtip = np.mat(tip)

m= np.shape(mbill)[1]
one = np.mat(np1.ones(m))
X = np.hstack((one.T,mbill.T))
```

```
#set k here
ypred = localWeightRegression(X,mtip,0.5)
SortIndex = X[:,1].argsort(0)
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();
```

Output



Additional Notes

Additional Notes

Additional Notes