SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Recognized by AICTE and Accredited by NBA, NAAC and an ISO 9001-2008 Certified Institution]

Bengaluru – 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING LABORATORY

CHOICE BASEDCREDITSYSTEM

15CSL76- VII Semester B.E

[Academic Year 2018-19]

Name : _____

USN : **Branch:** _____

_____

Semester : **Section:**

_____ _____

Compiled and Prepared by:

Ms. Ajina A & Ms. Mayuri
Assistant Professor
Dept. of CSE

Under the Guidance of:

Dr. G. C. Bhanu Prakash
Professor. & Head
Dept. of CSE

Department Vision and Mission

VISION

To create a center for imparting quality technical education and conducting research to meet the current and future challenges in the domain of computer science and engineering.

MISSION

- The Computer Science and Engineering department strives for excellence in teaching, applying, promoting and imparting knowledge through comprehensive academic programs.
- Train students to effectively apply the knowledge to solve real-world problems, thus enhance their potential for life-long high-quality career and give them a competitive advantage in the ever-changing and fast paced computing world.
- Prepare students to demonstrate a sense of societal and ethical responsibilities in their professional endeavors.
- Creating amongst students and faculty a collaborative environment open to the free exchange of ideas, where research, creativity, innovation and entrepreneurship can flourish.

PROGRAM OUTCOMES

PO's	PO Description
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**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.
PO3	**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.
PO4	**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.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**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.
PO7	**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.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**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.
PO11	**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.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO's	PSO Description
PSO1	An ability to apply the knowledge of mathematics, science, engineering fundamentals and specialization to analyze, design and implement complex computer based systems
PSO2	An ability to investigate a problem by analysis, interpretation of data, design of experiments, implementation through appropriate techniques, skills and tools to provide valid conclusions
PSO3	Be able to imply the engineering and management principles to function effectively as an individual, member or leader in diverse teams and multidisciplinary settings and also to communicate effectively on technical subject matters
PSO4	An understanding of professional, ethical, legal, security issues, societal responsibilities and indulge in life-long learning

SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Recognized by AICTE and Accredited by NBA, NAAC and an ISO 9001-2008 Certified Institution)

Bengaluru – 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Mr/Ms. _____ bearing USN _____ has completed the **MACHINE LEARNING LABORATORY** prescribed by Visvesvaraya Technology University for Seventh Semester, academic year 20__ - 20__

Name and Signature of Staff

Signature of HOD

Scheme and Syllabus

MACHINE LEARNING LABORATORY

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2018 -2019)

SEMESTER - VII

Laboratory Code	15CSL76	IA Marks	20
Number of Lecture Hours/Week	01I + 02P	Exam Marks	80
Total Number of Lecture Hours	40	Exam Hours	3

CREDITS - 02

Description (If any):

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

Laboratory Experiments :

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.
2. 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.
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.
4. Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.
5. Write a program to implement the **naïve Bayesianclassifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.
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.
7. Write a program to construct a **Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.
8. 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.

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..

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

Course objectives: This course will enable students to

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.

Study Experiment / Project:

Course outcomes: The students should be able to:

1. Understand the implementation procedures for the machine learning algorithms.
2. Design Java/Python programs for various Learning algorithms.
3. Apply appropriate data sets to the Machine Learning algorithms.
4. Identify and apply Machine Learning algorithms to solve real world problems.

Conduction of Practical Examination:

- All laboratory experiments are to be included for practical examination. · Students are allowed to pick one experiment from the lot.
- Strictly follow the instructions as printed on the cover page of answer script · Marks distribution: Procedure + Conduction + Viva:20 + 50 +10 (80)
- Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

Prerequisites

- Programming experience in Python
- Knowledge of basic Machine Learning Algorithms
- Knowledge of common statistical methods and data analysis best practices.

Software Requirements

1. Python version 3.5 and above
2. Machine Learning packages
   - Scikit-Learn
   - Numpy - matrices and linear algebra
   - Scipy - many numerical routines
   - Matplotlib- creating plots of data
   - Pandas –facilitates structured/tabular data manipulation and visualisations
   - Pomegranate –for fast and flexible probabilistic models
3. An Integrated Development Environment (IDE) for Python Programming

Anaconda

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages. Anaconda is used for scientific computing, data science, statistical analysis, and machine learning

Operating System

- Windows/Linux
- Anaconda Python distribution is compatible with Linux or windows.

Machine learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.

Machine learning tasks

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:

Supervised learning:

The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback:

Semi-supervised learning:

The computer is given only an incomplete training signal: a training set with some (often

many) of the target outputs missing.

Active learning:

The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labelling.

Reinforcement learning:

Training data (in form of rewards and punishments) is given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

Unsupervised learning:

No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Supervised learning	Un Supervised learning	Instance based learning
Find-s algorithm	EM algorithm	Locally weighted Regression algorithm
Candidate elimination algorithm	K means algorithm	
Decision tree algorithm		
Back propagation Algorithm		
Naïve Bayes Algorithm		
K nearest neighbour algorithm(lazy learning algorithm)		

Machine learning applications

In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised manner. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".In **regression,**also a supervised problem, the outputs are continuous rather than discrete.

Inclustering,a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

Density estimation findsthe distribution of inputs in some space.

Dimensionality reduction simplifiesinputs by mapping them into a lower-dimensional space. Topic modelling is a related problem, where a program is given a list of human language documents and is tasked with finding out which documents cover similar topics.

Machine learning Approaches

Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

Deep learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing.

Inductive logic programming

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

Support vector machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

Clustering

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some pre designated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning

Reinforcement learning

Reinforcement learning is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

Similarity and metric learning

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

Genetic algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

Rule-based machine learning

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply, knowledge. The defining characteristic of a rule-based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learners that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

Feature selection approach

Feature selection is the process of selecting an optimal subset of relevant features for use in model construction. It is assumed the data contains some features that are either redundant or irrelevant, and can thus be removed to reduce calculation cost without incurring much loss of information. Common optimality criteria include accuracy, similarity and information measures.

INDEX

S.NO	DATE	EXPRIMENT TITLE	Page Number	Marks allotted
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.		
2.		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.		
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.		
4.		Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.		
5.		Write a program to implement the **naïve Bayesianclassifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.		
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.		
7.		Write a program to construct a **Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.		
8.		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.		
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..		
10.		Implement the non-parametric **Locally Weighted Regression algorithm** in order to fit data points. Select appropriate data set for your experiment and draw graphs.		

Program No. 1

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.

Objective	To find most specific hypothesis in set of hypothesis which is consistent with positive training example?
Dataset	Tennis data set: This data set contains the set of example days on which playing of tennis is possible or not, based on attributes Sky, Air-Temp, Humidity, Wind, Water and Forecast.
ML algorithm	Supervised Learning-Find −S algorithm
Description	The FIND-S algorithm is probably one of the most simple machine learning algorithms.

Program

```
import random
import csv
def read_data(filename):
    with open(filename,"r") as csvfile:
        datareader=csv.reader(csvfile,delimiter=",")
        traindata=[]
        for row in datareader:
            traindata.append(row)
        return(traindata)

def findS(): #function for finding maximally specific set
  dataarr=read_data('ENJOYSPORT.csv')
  h=['0','0','0','0','0','0']
  rows=len(dataarr)
  columns=7
  for x in range(1,rows):
    t=dataarr[x]
    print(t)
    if t[columns-1]=='1':
      for y in range(0,columns-1):
        if h[y]==t[y]:
                pass
        elif h[y]!=t[y] and h[y]=='0':
                h[y]=t[y]
        elif h[y]!=t[y] and h[y]!='0':
                h[y]='?'
      print(h)
```

```
print("Maximally Specific set")
print('<',end=' ')
  for i in range(0,len(h)):
        print(h[i],',',end=' ')
  print('>')
findS()
```

dataset (ENJOYSPORT.csv)

```
Sky,AirTemp,Humidity,Wind,Water,Forecast,EnjoySport
Sunny,Warm,Normal,Strong,Warm,Same,1
Sunny,Warm,High,Strong,Warm,Same,1
Rainy,Cold,High,Strong,Warm,Change,0
Sunny,Warm,High,Strong,Cool,Change,1
```

output:

```
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', '1']
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', '1']
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', '0']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', '1']
['Sunny', 'Warm', '?', 'Strong', '?', '?']

Maximally Specific set
< Sunny , Warm , ? , Strong , ? , ? , >
```

Program No. 2

2. 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.

Objective	To find the set of all hypothesis consistent with both positive and negative training examples.
Dataset	Tennis data set: This data set contain the set of example days on which playing of tennis is possible or not. Based on attribute Sky, Air-Temp, Humidity, Wind, Water and Forecast. The dataset has 14 instances.
ML algorithm	Supervised Learning-Candidate elimination algorithm
Description	The **candidate elimination algorithm** incrementally builds the version space given a hypothesis space H and a set E of examples. Each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example.

Program

```
import csv
a = []
print("\n The Given Training Data Set \n")

with open('ws.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for row in reader:
        a.append (row)
        print(row)
num_attributes = len(a[0])-1

print("\n The initial value of hypothesis: ")
S = ['0'] * num_attributes
G = ['?'] * num_attributes
print ("\n The most specific hypothesis S0 : [0,0,0,0,0,0]\n")
print (" \n The most general hypothesis G0 : [?,?,?,?,?,?]\n")

# Comparing with First Training Example
for j in range(0,num_attributes):
        S[j] = a[0][j];

# Comparing with Remaining Training Examples of Given Data Set

print("\n Candidate Elimination algorithm  Hypotheses Version Space Computation\n")
temp=[]

for i in range(0,len(a)):
    print("-------------------------------------------------------------------------------")
    if a[i][num_attributes]=='yes':
        for j in range(0,num_attributes):
```

```
            if a[i][j]!=S[j]:
                S[j]='?'

        for j in range(0,num_attributes):
            for k  in range(1,len(temp)):
                if temp[k][j]!= '?' and temp[k][j] !=S[j]:
                    del temp[k]

        print(" For Training Example No :{0} the hypothesis is S{0}  ".format(i+1),S)
        if (len(temp)==0):
            print(" For Training Example No :{0} the hypothesis is G{0} ".format(i+1),G)
        else:
            print(" For Training Example No :{0} the hypothesis is G{0}".format(i+1),temp)

    if a[i][num_attributes]=='No':
        for j in range(0,num_attributes):
            if S[j] != a[i][j] and S[j]!= '?':
                G[j]=S[j]
                temp.append(G)
                G = ['?'] * num_attributes

        print(" For Training Example No :{0} the hypothesis is S{0} ".format(i+1),S)
        print(" For Training Example No :{0} the hypothesis is G{0}".format(i+1),temp)
```

Data Set: enjoysport.csv

```
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
```

Output:

```
The Given Training Data Set

['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 initial value of hypothesis:
 The most specific hypothesis S0 : [0,0,0,0,0,0]
 The most general hypothesis G0 : [?,?,?,?,?,?]

 Candidate Elimination algorithm  Hypotheses Version Space Computation

-------------------------------------------------------------------------------
 For Training Example No :1 the hypothesis is S1  ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm',
'Same']
 For Training Example No :1 the hypothesis is G1  ['?', '?', '?', '?', '?', '?']
-------------------------------------------------------------------------------
 For Training Example No :2 the hypothesis is S2  ['Sunny', 'Warm', '?', 'Strong', 'Warm',
'Same']
 For Training Example No :2 the hypothesis is G2  ['?', '?', '?', '?', '?', '?']
-------------------------------------------------------------------------------
 For Training Example No :3 the hypothesis is S3  ['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
 For Training Example No :3 the hypothesis is G3 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]
-------------------------------------------------------------------------------
 For Training Example No :4 the hypothesis is S4  ['Sunny', 'Warm', '?', 'Strong', '?', '?']
 For Training Example No :4 the hypothesis is G4 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?',
'?', '?', '?']]
```

Program No. 3

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.

Objective	To demonstrate the working of the decision tree based ID3 algorithm.
Dataset	Tennis data set: This data set contain the set of example days on which playing of tennis is possible or not. Based on attribute Sky, Air-Temp, Humidity, Wind, Water and Forecast.
ML algorithm	Supervised Learning-Decision Tree algorithm
Description	Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Program

```
import numpy as np
import math
from data_loader import read_data

class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""
    def __str__(self):
        return self.attribute

def subtables(data, col, delete):
    dict = {}
    items = np.unique(data[:, col])
    count = np.zeros((items.shape[0], 1), dtype=np.int32)
    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                count[x] += 1
    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="\S32")
        pos = 0
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos += 1
```

```
        if delete:
            dict[items[x]] = np.delete(dict[items[x]], col, 1)
    return items, dict

def entropy(S):
    items = np.unique(S)
    if items.size == 1:
        return 0
    counts = np.zeros((items.shape[0], 1))
    sums = 0
    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)
    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums

def gain_ratio(data, col):
    items, dict = subtables(data, col, delete=False)
    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))
    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0]/(total_size * 1.0)
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])
        intrinsic[x] = ratio * math.log(ratio, 2)
    total_entropy = entropy(data[:, -1])
    iv = -1 * sum(intrinsic)
    for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]
    return total_entropy / iv

def create_node(data, metadata):
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node
    gains = np.zeros((data.shape[1] - 1, 1))
    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)
    split = np.argmax(gains)
    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)
    items, dict = subtables(data, split, delete=True)
    for x in range(items.shape[0]):
```

```
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))
    return node


def empty(size):
    s = ""
    for x in range(size):
        s += " "
    return s

def print_tree(node, level):
    if node.answer != "":
        print(empty(level), node.answer)
        return
    print(empty(level), node.attribute)
    for value, n in node.children:
        print(empty(level + 1), value)
        print_tree(n, level + 2)

metadata, traindata = read_data("tennis.csv")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
```

dataloader.py

```
def read_data(filename):
    with open(filename,'r') as csvfile:
        datareader = csv.reader(csvfile,delimiter=',')
        headers = next(datareader)
        metadata=[]
        traindata=[]
        for name in headers:
            metadata.append(name)
        for row in datareader:
            traindata.append(row)
    return(metadata,traindata)
```

Data Set(Tennis.csv)

```
outlook, temperature, humidity, wind, answer
sunny, hot, high, weak, no
sunny, hot, high, strong, no
overcast, hot, high, weak, yes
rain, mild, high, weak, yes
rain, cool, normal, weak, yes
rain, cool, normal, strong, no
overcast, cool, normal, strong, yes
sunny, mild, high, weak, no
sunny, cool, normal, weak, yes
rain, mild, normal, weak, yes
sunny, mild, normal, strong, yes
overcast, mild, high, strong, yes
overcast, hot, normal, weak, yes
rain, mild, high, strong, no
```

OUTPUT:

```
outlook
        overcast
                b'yes'
        rain
                wind
                        b'strong'
                                b'no'
                        b'weak'
                                b'yes'
        sunny
                humidity
                        b'high'
                                b'no'
                        b'normal'
                                b'yes
```

Program No. 4

4. Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

Objective	To build an artificial neural network using the back propagation algorithm.
Dataset	Data stored as a list having two features- number of hours slept, number of hours studied with the test score being the class label
ML algorithm	Supervised Learning –Back propagation algorithm
Description	The neural network using back propagation will model a single hidden layer with three inputs and one output. The network will be predicting the score of an exam based on the inputs of number of hours studied and the number of hours slept the day before. The test score is the output.

Program

```
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=7000 #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 err
        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("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:
 [[0.89440433]
 [0.88100177]
 [0.89441761]]
```

Program No. 5

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.

Objective	To implement a classification model for sample training dataset and computing the accuracy of the classifier for test data.
Dataset	Pima Indian Diabetes dataset stored as .CSV .The attributes are Number of times pregnant, Plasma glucose concentration, Blood Pressure, Triceps skin fold thickness, serum insulin, Body mass index, Diabetes pedigree function, Age
ML algorithm	Supervised Learning -Naïve Bayes Algorithm
Description	The Naïve Bayes classifier is a probabilistic classifier that is based on Bayes Theorem. The algorithm builds a model assuming that the attributes in the dataset are independent of each other.

Program

```
#https://www.kaggle.com/ilango/naive-bayes-from-the-scratch

import numpy as np
import pandas as pd
mush = pd.read_csv("flu.csv")
mush.replace('?',np.nan,inplace=True)
print(len(mush.columns),"columns, after dropping NA,",len(mush.dropna(axis=1).columns))

#drop wherever you have ? the values are not known
mush.dropna(axis=1,inplace=True)

#the first column in dataset is class which is target variable
target = 'flu'
features = mush.columns[mush.columns != target]
classes = mush[target].unique()
test = mush.sample(frac=0.3)
mush = mush.drop(test.index)
probs = {}
probcl = {}

for x in classes:
    mushcl = mush[mush[target]==x][features]
    clsp = {}
    tot = len(mushcl)
    for col in mushcl.columns:
        colp = {}
        for val,cnt in mushcl[col].value_counts().iteritems():
         #df = pd.DataFrame({'mycolumn': [1,2,2,2,3,3,4]})
          #for val, cnt in df.mycolumn.value_counts().iteritems():
         # print 'value', val, 'was found', cnt, 'times'
          # value 2 was found 3 times
          #value 3 was found 2 times
           #value 4 was found 1 times
```

```
            #value 1 was found 1 times

            pr = cnt/tot
            colp[val] = pr
            clsp[col] = colp

    probs[x] = clsp
    probcl[x] = len(mushcl)/len(mush)


def probabs(x):
    #X - pandas Series with index as feature
    if not isinstance(x,pd.Series):
        raise IOError("Arg must of type Series")
    probab = {}

  for cl in classes:
        pr = probcl[cl]
        for col,val in x.iteritems():
            try:
                pr *= probs[cl][col][val]
            except KeyError:
                pr = 0
        probab[cl] = pr
    return probab

def classify(x):
    probab = probabs(x)
    mx = 0
    mxcl = ''
    for cl,pr in probab.items():
        if pr > mx:
            mx = pr
            mxcl = cl
    return mxcl

#Train data
b = []
for i in mush.index:
   # print(classify(mush.loc[i,features]),mush.loc[i,target])
    b.append(classify(mush.loc[i,features]) == mush.loc[i,target])
print(sum(b),"correct of",len(mush))
print("Accuracy:", sum(b)/len(mush))
#Test data
b = []
for i in test.index:
    #print(classify(mush.loc[i,features]),mush.loc[i,target])
    b.append(classify(test.loc[i,features]) == test.loc[i,target])
print(sum(b),"correct of",len(test))
print("Accuracy:",sum(b)/len(test))
```

DATASET

mush.csv

class,cap-shape,cap-surface,cap-color,bruises,odor,gill-attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-above-ring,stalk-color-below-ring,veil-type,veil-color,ring-number,ring-type

p,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p
e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p
e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p
e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p
e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,x,f,n,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e
e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,f,f,w,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e
p,x,s,n,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p
p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,b,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p
p,x,y,n,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p
e,b,y,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,b,y,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,b,s,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p
p,f,s,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,x,y,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,f,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,x,s,y,t,a,f,w,n,n,t,b,s,s,w,w,p,w,o,p
e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,x,y,y,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,x,y,n,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p
e,b,y,y,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,x,f,y,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p
e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p
p,x,y,n,t,p,f,c,n,w,e,e,s,s,w,w,p,w,o,p
e,x,f,y,t,a,f,w,n,p,t,b,s,s,w,w,p,w,o,p
e,b,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,b,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,x,y,y,t,l,f,c,b,n,e,r,s,y,w,w,p,w,o,p

```
e,x,f,n,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p
e,x,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,x,y,w,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p
e,x,y,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,x,s,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,x,y,y,t,l,f,c,b,n,e,r,s,y,w,w,p,w,o,p
e,f,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p
e,x,y,n,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p
e,x,s,w,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,b,s,w,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
p,x,y,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
p,x,s,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,b,y,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,f,f,g,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e
e,b,s,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p
e,x,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,x,y,n,t,a,f,c,b,p,e,r,s,y,w,w,p,w,o,p
e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p
e,b,y,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p
e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p
e,b,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p
e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p
```

OUTPUT:

```
5 columns, after dropping NA, 5
6 correct of 6
Accuracy: 1.0
1 correct of 2
Accuracy: 0.5
```

Program No. 6

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.

Objective	To implement a binary classification model that classifies a set of documents and calculates the accuracy, precision and recall for the dataset
Dataset	Contains text as sentences labelled positive and negative. The dataset contains a total of 10 instances.
ML algorithm	Supervised Learning -Naïve Bayes Algorithm
Packages	Scikit learn(sklearn),pandas
Description	The Naïve Bayes classifier is a probabilistic classifier that is based on Bayes Theorem. The algorithm builds a model assuming that the attributes in the dataset are independent of each other.

Program

```
import pandas as pd
msg=pd.read_csv('naivetrext1.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
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print(xtest.shape)
print(xtrain.shape)
print(ytest.shape)
print(ytrain.shape)
print("train data")
print(xtrain)
#output of count vectoriser is a sparse matrix
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print(count_vect.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())
print(df)#tabular representation
print(xtrain_dtm) #sparse matrix representation

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

#printing accuracy metrics
```

```
from sklearn import metrics
print('Accuracy metrics')
print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
print('Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('Recall and Precison ')
print(metrics.recall_score(ytest,predicted))
print(metrics.precision_score(ytest,predicted))
'''docs_new = ['I like this place', 'My boss is not my saviour']
X_new_counts = count_vect.transform(docs_new)
predictednew = clf.predict(X_new_counts)
for doc, category in zip(docs_new, predictednew):
print('%s->%s' % (doc, msg.labelnum[category]))'''
```

DATASET

```
I love this sandwich, pos
This is an amazing place, pos
I feel very good about these beers, pos
This is my best work, pos
What an awesome view, pos
I do not like this restaurant, neg
I am tired of this stuff, neg
I can't deal with this, neg
He is my sworn enemy, neg
My boss is horrible, neg
This is an awesome place, pos
I do not like the taste of this juice, neg
I love to dance, pos
I am sick and tired of this place, neg
What a great holiday, pos
That is a bad locality to stay, neg
We will have good fun tomorrow, pos
I went to my enemy's house today, neg
```

OUTPUT:

```
['about', 'am', 'amazing', 'an', 'and', 'awesome', 'beers', 'best', 'boss', 'can', 'deal', 'do', 'enemy',
'feel', 'fun', 'good', 'great', 'have', 'holiday', 'horrible', 'house', 'is', 'juice', 'like', 'my', 'not', 'of',
'place', 'restaurant', 'sick', 'stuff', 'taste', 'the', 'these', 'this', 'tired', 'to', 'today', 'tomorrow',
'very', 'we', 'went', 'what', 'will', 'with', 'work']
   about  am  amazing  an  and  awesome  ...  we  went  what  will  with  work
0      0   1        0   0    0        0  ...   0     0     0     0     0     0
1      0   0        0   0    0        0  ...   0     1     0     0     0     0
2      0   1        0   0    1        0  ...   0     0     0     0     0     0
3      0   0        0   0    0        0  ...   1     0     0     1     0     0
4      1   0        0   0    0        0  ...   0     0     0     0     0     0
5      0   0        0   0    0        0  ...   0     0     1     0     0     0
6      0   0        0   0    0        0  ...   0     0     0     0     0     1
7      0   0        0   1    0        1  ...   0     0     0     0     0     0
```

```
8     0  0     0  0  0     0 ...  0  0  0  0  0  0
9     0  0     0  0  0     0 ...  0  0  0  0  0  0
10    0  0     0  0  0     0 ...  0  0  0  0  1  0
11    0  0     1  1  0     0 ...  0  0  0  0  0  0
12    0  0     0  0  0     0 ...  0  0  0  0  0  0
Accuracy metrics
Accuracy of the classifer is 0.8
Confusion matrix
[[2 0]
 [1 2]]
Recall and Precison
0.6666666666666666
1.0
```

Program No. 7

7. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

Objective	To construct a Bayesian network considering medical data.
Dataset	Heart.CSV
ML algorithm	Naïve Bayes Model on conditional probability
Packages	
Description	

Program

```
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

# Read the attributes
lines = list(csv.reader(open('data/heart.csv', 'r')));
attributes = lines[0]

#attributes = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang','oldpeak',
'slope', 'ca', 'thal', 'heartdisease']
# Read Cleveland Heart disease data
heartDisease = pd.read_csv('data/heart.csv', names = attributes)
heartDisease = heartDisease.replace('?', np.nan)

# Display the data
print('Few examples from the dataset are given below')
print(heartDisease.head())
print('\nAttributes and datatypes')
print(heartDisease.dtypes)

# Model Bayesian Network
model = BayesianModel([ ('age', 'trestbps'), ('age', 'fbs'), ('sex', 'trestbps'),
```

```
                ('sex', 'trestbps'), ('exang', 'trestbps'), ('trestbps', 'heartdisease'),
                ('fbs', 'heartdisease'), ('heartdisease', 'restecg'), ('heartdisease', 'thalach'),
                ('heartdisease', 'chol')])
# Learning CPDs using Maximum Likelihood Estimators
print('\nLearning CPDs using Maximum Likelihood Estimators...')
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)

# Deducing with Bayesian Network
print('\nInferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)

print('\n1.Probability of HeartDisease given Age=20')
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'age': 40})

print(q['heartdisease'])

print('\n2. Probability of HeartDisease given chol (Cholestoral) =100')

q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'sex': 0, 'chol': 100})
print(q['heartdisease'])
```

DataSet:

```
age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,heartdisease
63,1,1,145,233,1,2,150,0,2.3,3,0,6,0
67,1,4,160,286,0,2,108,1,1.5,2,3,3,2
67,1,4,120,229,0,2,129,1,2.6,2,2,7,1
37,1,3,130,250,0,0,187,0,3.5,3,0,3,0
41,0,2,130,204,0,2,172,0,1.4,1,0,3,0
56,1,2,120,236,0,0,178,0,0.8,1,0,3,0
62,0,4,140,268,0,2,160,0,3.6,3,2,3,3
57,0,4,120,354,0,0,163,1,0.6,1,0,3,0
63,1,4,130,254,0,2,147,0,1.4,2,1,7,2
53,1,4,140,203,1,2,155,1,3.1,3,0,7,1
57,1,4,140,192,0,0,148,0,0.4,2,0,6,0
56,0,2,140,294,0,2,153,0,1.3,2,0,3,0
56,1,3,130,256,1,2,142,1,0.6,2,1,6,2
44,1,2,120,263,0,0,173,0,0,1,0,7,0
52,1,3,172,199,1,0,162,0,0.5,1,0,7,0
57,1,3,150,168,0,0,174,0,1.6,1,0,3,0
48,1,2,110,229,0,0,168,0,1,3,0,7,1
54,1,4,140,239,0,0,160,0,1.2,1,0,3,0
48,0,3,130,275,0,0,139,0,0.2,1,0,3,0
49,1,2,130,266,0,0,171,0,0.6,1,0,3,0
```

```
64,1,1,110,211,0,2,144,1,1.8,2,0,3,0
58,0,1,150,283,1,2,162,0,1,1,0,3,0
58,1,2,120,284,0,2,160,0,1.8,2,0,3,1
58,1,3,132,224,0,2,173,0,3.2,1,2,7,3
60,1,4,130,206,0,2,132,1,2.4,2,2,7,4
50,0,3,120,219,0,0,158,0,1.6,2,0,3,0
58,0,3,120,340,0,0,172,0,0,1,0,3,0
66,0,1,150,226,0,0,114,0,2.6,3,0,3,0
43,1,4,150,247,0,0,171,0,1.5,1,0,3,0
40,1,4,110,167,0,2,114,1,2,2,0,7,3
69,0,1,140,239,0,0,151,0,1.8,1,2,3,0
60,1,4,117,230,1,0,160,1,1.4,1,2,7,2
64,1,3,140,335,0,0,158,0,0,1,0,3,1
59,1,4,135,234,0,0,161,0,0.5,2,0,7,0
44,1,3,130,233,0,0,179,1,0.4,1,0,3,0
42,1,4,140,226,0,0,178,0,0,1,0,3,0
43,1,4,120,177,0,2,120,1,2.5,2,0,7,3
57,1,4,150,276,0,2,112,1,0.6,2,1,6,1
55,1,4,132,353,0,0,132,1,1.2,2,1,7,3
61,1,3,150,243,1,0,137,1,1,2,0,3,0
65,0,4,150,225,0,2,114,0,1,2,3,7,4
40,1,1,140,199,0,0,178,1,1.4,1,0,7,0
71,0,2,160,302,0,0,162,0,0.4,1,2,3,0
59,1,3,150,212,1,0,157,0,1.6,1,0,3,0
61,0,4,130,330,0,2,169,0,0,1,0,3,1
58,1,3,112,230,0,2,165,0,2.5,2,1,7,4
51,1,3,110,175,0,0,123,0,0.6,1,0,3,0
50,1,4,150,243,0,2,128,0,2.6,2,0,7,4
65,0,3,140,417,1,2,157,0,0.8,1,1,3,0
53,1,3,130,197,1,2,152,0,1.2,3,0,3,0
41,0,2,105,198,0,0,168,0,0,1,1,3,0
65,1,4,120,177,0,0,140,0,0.4,1,0,7,0
```

OUTPUT:

Program No. 8

8. 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.

Objective	To group a set of unlabelled data into similar classes/clusters and label them and to compare the quality of algorithm.
Dataset	Delivery fleet driver dataset Data set in .csv file with features "Driver_ID", "distance_feature "," speeding_feature" having more than 20 instances
ML algorithm	EM algorithm, K means algorithm – Unsupervised clustering
Packages	Scikit learn(sklearn),pandas
Description	***EM algorithm*** – soft clustering - can be used for variable whose value is never directly observed, provided the general probability distribution governing these variables is known. EM algorithm can be used to train Bayesian belief networks as well as radial basis function network. ***K-Means*** – Hard Clustering - to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

Program

```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
# Importing the dataset
data = pd.read_csv('ex.csv')
print("Input Data and Shape")
print(data.shape)
data.head()
print(data.head())

# Getting the values and plotting it
f1 = data['V1'].values
print("f1")
print(f1)
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
print("x")
```

```
print(X)
print('Graph for whole dataset')
plt.scatter(f1, f2, c='black', s=600)
plt.show()
##############################################
kmeans = KMeans(2, random_state=0)
labels = kmeans.fit(X).predict(X)
print("labels")
print(labels)
centroids = kmeans.cluster_centers_
print("centroids")
print(centroids)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=40);
print('Graph using Kmeans Algorithm')
plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=200, c='#050505')
plt.show()
#gmm demo
gmm = GaussianMixture(n_components=2).fit(X)
labels = gmm.predict(X)
print("ILABELS GMM")
print(labels)
probs = gmm.predict_proba(X)
size = 10 * probs.max(1) ** 3
print('Graph using EM Algorithm')
#print(probs[:300].round(4))
plt.scatter(X[:, 0], X[:, 1], c=labels, s=size, cmap='viridis');
plt.show()
```

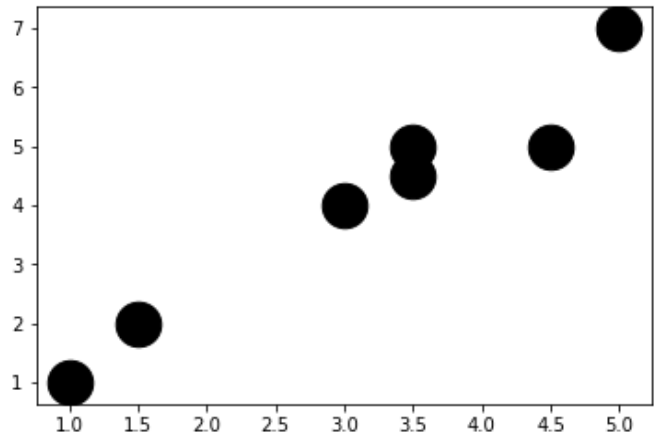
ex.csv

```
",V1,V2
1,1,1
2,1.5,2
3,3,4
4,5,7
5,3.5,5
6,4.5,5
7,3.5,4.5
```

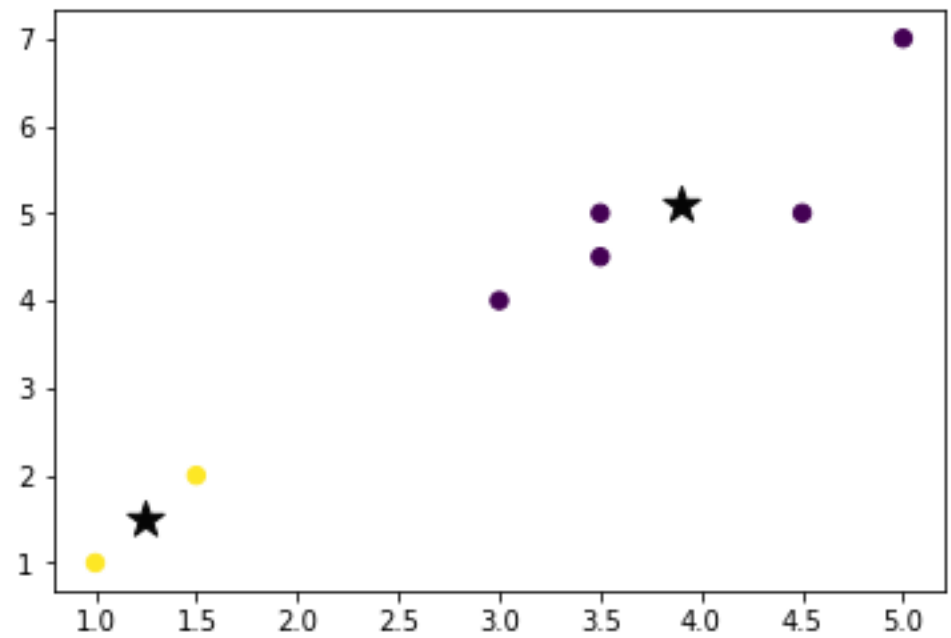
OUTPUT

```
Input Data and Shape
(7, 3)
   ""   V1   V2
0   1  1.0  1.0
1   2  1.5  2.0
2   3  3.0  4.0
3   4  5.0  7.0
4   5  3.5  5.0
f1
[1.  1.5 3.  5.  3.5 4.5 3.5]
```

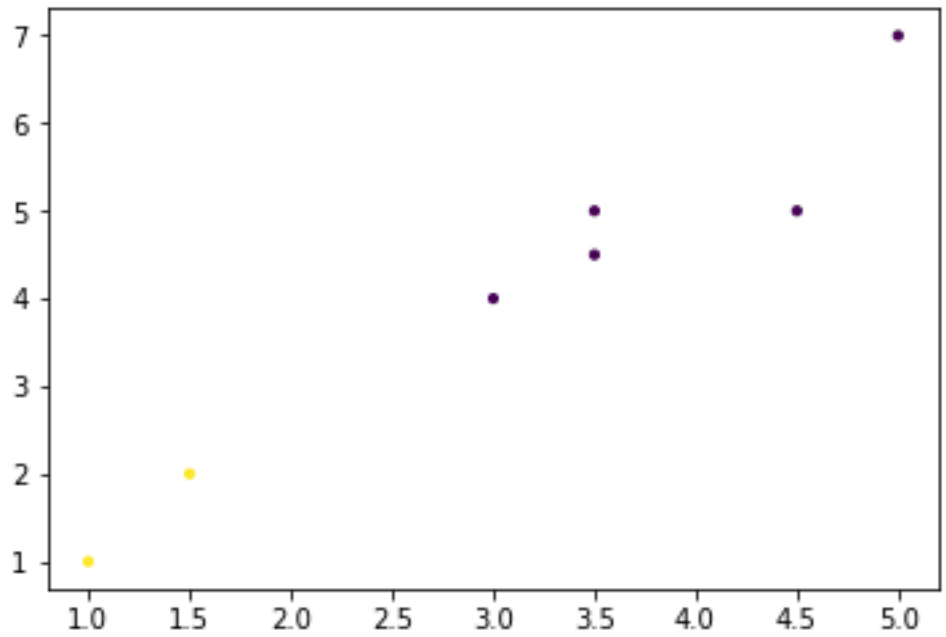
x
[[1. 1.]
[1.5 2.]
[3. 4.]
[5. 7.]
[3.5 5.]
[4.5 5.]
[3.5 4.5]]
Graph for whole dataset



labels
[1 1 0 0 0 0 0]
centroids
[[3.9 5.1]
[1.25 1.5]]
Graph using K- Means algorithm



LABELS GMM
[1 1 0 0 0 0 0]
Graph using EM Algorithm



Program No. 9

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..

Objective	To implement a classification model that classifies a set of documents and calculates the accuracy, precision and recall for the dataset
Dataset	IRIS data set with features "petal_length", "petal_width", "sepal_length", "sepal_width" having more than 150 instances
ML algorithm	Supervised Learning – Lazy learning algorithm
Packages	Scikit learn(sklearn),pandas
Description	When we get training set/instances, machine won't learn or a model can't be built. Instead instances/examples will be just stored in memory. Test instance is given; attempt to find the closest instance/most neighbouring instances in the instance space

Program:

```
from sklearn.datasets import load_iris

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

from sklearn.model_selection import train_test_split

iris_dataset=load_iris()

print("\n IRIS FEATURES \ TARGET NAMES: \n ", iris_dataset.target_names)

for i in range(len(iris_dataset.target_names)):

    print("\n[{0}]:[{1}]".format(i,iris_dataset.target_names[i]))


print("\n IRIS DATA :\n",iris_dataset["data"])


X_train, X_test, y_train, y_test = train_test_split(iris_dataset["data"], iris_dataset["target"],
random_state=0)


print("\n Target :\n",iris_dataset["target"])

print("\n X TRAIN \n", X_train)

print("\n X TEST \n", X_test)

print("\n Y TRAIN \n", y_train)
```

```
print("\n Y TEST \n", y_test)

kn = KNeighborsClassifier(n_neighbors=1)

kn.fit(X_train, y_train)



for i in range(len(X_test)):

    x = X_test[i]

    x_new = np.array([x])

    prediction = kn.predict(x_new)

    print("\n Actual : {0}{1}, Predicted

:{2}{3}".format(y_test[i],iris_dataset["target_names"][y_test[i]],prediction,iris_dataset["targe
t_names"][prediction]))



print("\n TEST SCORE[ACCURACY]: {:.2f}\n".format(kn.score(X_test, y_test)))
```

OUTPUT:

```
IRIS FEATURES \ TARGET NAMES:
 ['setosa' 'versicolor' 'virginica']
```

```
[0]:[setosa]

[1]:[versicolor]

[2]:[virginica]

IRIS DATA :
[[5.1 3.5 1.4 0.2]
[4.9 3.  1.4 0.2]
[4.7 3.2 1.3 0.2]
[4.6 3.1 1.5 0.2]
[5.  3.6 1.4 0.2]
[5.4 3.9 1.7 0.4]
[4.6 3.4 1.4 0.3]
[5.  3.4 1.5 0.2]
[4.4 2.9 1.4 0.2]
[4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]

  .

  .

  .
Target :
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2]

X TRAIN
[[5.9 3.  4.2 1.5]
[5.8 2.6 4.  1.2]
[6.8 3.  5.5 2.1]
[4.7 3.2 1.3 0.2]
[6.9 3.1 5.1 2.3]
[5.  3.5 1.6 0.6]
[5.4 3.7 1.5 0.2]
[5.  2.  3.5 1. ]
[6.5 3.  5.5 1.8]
[6.7 3.3 5.7 2.5]
[6.  2.2 5.  1.5]
[6.7 2.5 5.8 1.8]
[5.6 2.5 3.9 1.1]
```

```
[7.7 3.  6.1 2.3]
[6.3 3.3 4.7 1.6]
[5.5 2.4 3.8 1.1]
[6.3 2.7 4.9 1.8]
[6.3 2.8 5.1 1.5]
[4.9 2.5 4.5 1.7]
[6.3 2.5 5.  1.9]
[7.  3.2 4.7 1.4]
[6.5 3.  5.2 2. ]
[6.  3.4 4.5 1.6]
[4.8 3.1 1.6 0.2]
[5.8 2.7 5.1 1.9]
[5.6 2.7 4.2 1.3]
[5.6 2.9 3.6 1.3]
[5.5 2.5 4.  1.3]
[6.1 3.  4.6 1.4]
[7.2 3.2 6.  1.8]
[5.3 3.7 1.5 0.2]
[4.3 3.  1.1 0.1]
[6.4 2.7 5.3 1.9]
[5.7 3.  4.2 1.2]
[5.4 3.4 1.7 0.2]
[5.7 4.4 1.5 0.4]
[6.9 3.1 4.9 1.5]
[4.6 3.1 1.5 0.2]
[5.9 3.  5.1 1.8]
[5.1 2.5 3.  1.1]
[4.6 3.4 1.4 0.3]
[6.2 2.2 4.5 1.5]
[7.2 3.6 6.1 2.5]
[5.7 2.9 4.2 1.3]
[4.8 3.  1.4 0.1]
[7.1 3.  5.9 2.1]
[6.9 3.2 5.7 2.3]
[6.5 3.  5.8 2.2]
[6.4 2.8 5.6 2.1]
[5.1 3.8 1.6 0.2]
[4.8 3.4 1.6 0.2]
[6.5 3.2 5.1 2. ]
[6.7 3.3 5.7 2.1]
[4.5 2.3 1.3 0.3]
[6.2 3.4 5.4 2.3]
[4.9 3.  1.4 0.2]
[5.7 2.5 5.  2. ]
[6.9 3.1 5.4 2.1]
[4.4 3.2 1.3 0.2]
[5.  3.6 1.4 0.2]
[7.2 3.  5.8 1.6]
[5.1 3.5 1.4 0.3]
[4.4 3.  1.3 0.2]
[5.4 3.9 1.7 0.4]
```

```
[5.5 2.3 4.  1.3]
[6.8 3.2 5.9 2.3]
[7.6 3.  6.6 2.1]
[5.1 3.5 1.4 0.2]
[4.9 3.1 1.5 0.1]
[5.2 3.4 1.4 0.2]
[5.7 2.8 4.5 1.3]
[6.6 3.  4.4 1.4]
[5.  3.2 1.2 0.2]
[5.1 3.3 1.7 0.5]
[6.4 2.9 4.3 1.3]
[5.4 3.4 1.5 0.4]
[7.7 2.6 6.9 2.3]
[4.9 2.4 3.3 1. ]
[7.9 3.8 6.4 2. ]
[6.7 3.1 4.4 1.4]
[5.2 4.1 1.5 0.1]
[6.  3.  4.8 1.8]
[5.8 4.  1.2 0.2]
[7.7 2.8 6.7 2. ]
[5.1 3.8 1.5 0.3]
[4.7 3.2 1.6 0.2]
[7.4 2.8 6.1 1.9]
[5.  3.3 1.4 0.2]
[6.3 3.4 5.6 2.4]
[5.7 2.8 4.1 1.3]
[5.8 2.7 3.9 1.2]
[5.7 2.6 3.5 1. ]
[6.4 3.2 5.3 2.3]
[6.7 3.  5.2 2.3]
[6.3 2.5 4.9 1.5]
[6.7 3.  5.  1.7]
[5.  3.  1.6 0.2]
[5.5 2.4 3.7 1. ]
[6.7 3.1 5.6 2.4]
[5.8 2.7 5.1 1.9]
[5.1 3.4 1.5 0.2]
[6.6 2.9 4.6 1.3]
[5.6 3.  4.1 1.3]
[5.9 3.2 4.8 1.8]
[6.3 2.3 4.4 1.3]
[5.5 3.5 1.3 0.2]
[5.1 3.7 1.5 0.4]
[4.9 3.1 1.5 0.1]
[6.3 2.9 5.6 1.8]
[5.8 2.7 4.1 1. ]
[7.7 3.8 6.7 2.2]
[4.6 3.2 1.4 0.2]]

X TEST
[[5.8 2.8 5.1 2.4]
```

```
[6.  2.2 4.  1. ]
[5.5 4.2 1.4 0.2]
[7.3 2.9 6.3 1.8]
[5.  3.4 1.5 0.2]
[6.3 3.3 6.  2.5]
[5.  3.5 1.3 0.3]
[6.7 3.1 4.7 1.5]
[6.8 2.8 4.8 1.4]
[6.1 2.8 4.  1.3]
[6.1 2.6 5.6 1.4]
[6.4 3.2 4.5 1.5]
[6.1 2.8 4.7 1.2]
[6.5 2.8 4.6 1.5]
[6.1 2.9 4.7 1.4]
[4.9 3.1 1.5 0.1]
[6.  2.9 4.5 1.5]
[5.5 2.6 4.4 1.2]
[4.8 3.  1.4 0.3]
[5.4 3.9 1.3 0.4]
[5.6 2.8 4.9 2. ]
[5.6 3.  4.5 1.5]
[4.8 3.4 1.9 0.2]
[4.4 2.9 1.4 0.2]
[6.2 2.8 4.8 1.8]
[4.6 3.6 1.  0.2]
[5.1 3.8 1.9 0.4]
[6.2 2.9 4.3 1.3]
[5.  2.3 3.3 1. ]
[5.  3.4 1.6 0.4]
[6.4 3.1 5.5 1.8]
[5.4 3.  4.5 1.5]
[5.2 3.5 1.5 0.2]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.2]
[5.2 2.7 3.9 1.4]
[5.7 3.8 1.7 0.3]
[6.  2.7 5.1 1.6]]

y TRAIN
[1 1 2 0 2 0 0 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 0 2 1 1 1 1 2 0 0 2 1 0 0 1
 0 2 1 0 1 2 1 0 2 2 2 2 0 0 2 2 0 2 0 2 2 0 0 2 0 0 0 1 2 2 0 0 0 1 1 0 0
 1 0 2 1 2 1 0 2 0 2 0 0 2 0 2 1 1 1 2 2 1 1 0 1 2 2 0 1 1 1 1 0 0 0 2 1 2
 0]

y TEST
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1]

Actual : 2 virginica, Predicted :[2]['virginica']

Actual : 1 versicolor, Predicted :[1]['versicolor']
```

```
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 2 virginica, Predicted :[2]['virginica']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 2 virginica, Predicted :[2]['virginica']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 2 virginica, Predicted :[2]['virginica']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 2 virginica, Predicted :[2]['virginica']
Actual : 1 versicolor, Predicted :[1]['versicolor']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 2 virginica, Predicted :[2]['virginica']
Actual : 0 setosa, Predicted :[0]['setosa']
Actual : 0 setosa, Predicted :[0]['setosa']
```

```
Actual : 1 versicolor, Predicted :[1]['versicolor']

Actual : 1 versicolor, Predicted :[1]['versicolor']

Actual : 0 setosa, Predicted :[0]['setosa']

Actual : 2 virginica, Predicted :[2]['virginica']

Actual : 1 versicolor, Predicted :[1]['versicolor']

Actual : 0 setosa, Predicted :[0]['setosa']

Actual : 2 virginica, Predicted :[2]['virginica']

Actual : 2 virginica, Predicted :[2]['virginica']

Actual : 1 versicolor, Predicted :[1]['versicolor']

Actual : 0 setosa, Predicted :[0]['setosa']

Actual : 1 versicolor, Predicted :[2]['virginica']

TEST SCORE[ACCURACY]: 0.97
```

Program No. 10

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

Objective	To implement Regression algorithm to fit the given data
Dataset	The dataset contains billing information based on the attributes total_bill, tip, sex, smoker, day, time, size
ML algorithm	Locally Weighted Regression Algorithm – Instance Based learning
Description	Regression means approximating a real valued target function. Given a new query instance X_q, the general approach is to construct an approximation function F that fits the training example in the neighbourhood surrounding X_q. This approximation is then used to estimate the target value $F(X_q)$

Program

```
from numpy import *
import operator
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from numpy.linalg import *
def kernel(point,xmat, k):
    m,n = shape(xmat)
    weights = mat(eye((m)))
    for j in range(m):
        diff = point - X[j]

        weights[j,j] = 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 = shape(xmat)

    ypred = zeros(m)
    for i in range(m):

        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred
data = pd.read_csv('tips.csv')
bill = array(data.total_bill)
tip = array(data.tip)
```

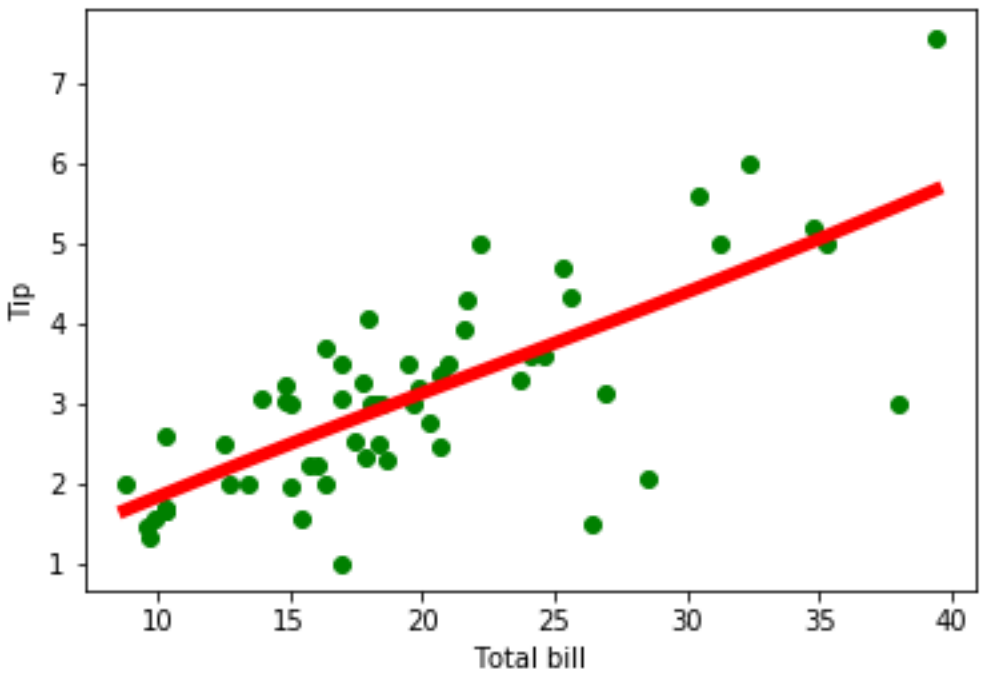
```
mbill = mat(bill)
mtip = mat(tip)
m= shape(mbill)[1]

one = mat(ones(m))

X= hstack((one.T,mbill.T))

#set k here
ypred = localWeightRegression(X,mtip,10)

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();
```

Data Set:

```
total_bill,tip,sex,smoker,day,time,size
16.99,1.01,Female,No,Sun,Dinner,2
10.34,1.66,Male,No,Sun,Dinner,3
21.01,3.5,Male,No,Sun,Dinner,3
23.68,3.31,Male,No,Sun,Dinner,2
24.59,3.61,Female,No,Sun,Dinner,4
25.29,4.71,Male,No,Sun,Dinner,4
8.77,2.0,Male,No,Sun,Dinner,2
26.88,3.12,Male,No,Sun,Dinner,4
15.04,1.96,Male,No,Sun,Dinner,2
14.78,3.23,Male,No,Sun,Dinner,2
10.27,1.71,Male,No,Sun,Dinner,2
35.26,5.0,Female,No,Sun,Dinner,4
15.42,1.57,Male,No,Sun,Dinner,2
18.43,3.0,Male,No,Sun,Dinner,4
14.83,3.02,Female,No,Sun,Dinner,2
21.58,3.92,Male,No,Sun,Dinner,2
10.33,1.67,Female,No,Sun,Dinner,3
16.29,3.71,Male,No,Sun,Dinner,3
16.97,3.5,Female,No,Sun,Dinner,3
20.65,3.35,Male,No,Sat,Dinner,3
17.92,4.08,Male,No,Sat,Dinner,2
20.29,2.75,Female,No,Sat,Dinner,2
15.77,2.23,Female,No,Sat,Dinner,2
39.42,7.58,Male,No,Sat,Dinner,4
19.82,3.18,Male,No,Sat,Dinner,2
17.81,2.34,Male,No,Sat,Dinner,4
13.37,2.0,Male,No,Sat,Dinner,2
12.69,2.0,Male,No,Sat,Dinner,2
21.7,4.3,Male,No,Sat,Dinner,2
19.65,3.0,Female,No,Sat,Dinner,2
9.55,1.45,Male,No,Sat,Dinner,2
18.35,2.5,Male,No,Sat,Dinner,4
15.06,3.0,Female,No,Sat,Dinner,2
20.69,2.45,Female,No,Sat,Dinner,4
17.78,3.27,Male,No,Sat,Dinner,2
24.06,3.6,Male,No,Sat,Dinner,3
16.31,2.0,Male,No,Sat,Dinner,3
16.93,3.07,Female,No,Sat,Dinner,3
18.69,2.31,Male,No,Sat,Dinner,3
31.27,5.0,Male,No,Sat,Dinner,3
16.04,2.24,Male,No,Sat,Dinner,3
17.46,2.54,Male,No,Sun,Dinner,2
13.94,3.06,Male,No,Sun,Dinner,2
9.68,1.32,Male,No,Sun,Dinner,2
30.4,5.6,Male,No,Sun,Dinner,4
18.29,3.0,Male,No,Sun,Dinner,2
22.23,5.0,Male,No,Sun,Dinner,2
32.4,6.0,Male,No,Sun,Dinner,4
28.55,2.05,Male,No,Sun,Dinner,3
```

```
18.04,3.0,Male,No,Sun,Dinner,2
12.54,2.5,Male,No,Sun,Dinner,2
10.29,2.6,Female,No,Sun,Dinner,2
34.81,5.2,Female,No,Sun,Dinner,4
9.94,1.56,Male,No,Sun,Dinner,2
25.56,4.34,Male,No,Sun,Dinner,4
19.49,3.51,Male,No,Sun,Dinner,2
38.01,3.0,Male,Yes,Sat,Dinner,4
26.41,1.5,Female,No,Sat,Dinner,2
```

OUTPUT:



SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY
[Affiliated to VTU, Recognized by AICTE and Accredited by NBA, NAAC and an ISO 9001-2008 Certified Institution)
Bengaluru – 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB EVALUATION

USN	:		SEMESTER / SECTION	:	7 /
STUDENT NAME	:				
SUBJECT NAME	:	Machine Learning Laboratory	SUBCODE	:	15CSL76

Present & Communicate Effectively			Understanding of Fundamental Knowledge			Interpretation & Analysis of result		
WEAK	MEDIUM	HIGH	WEAK	MEDIUM	HIGH	WEAK	MEDIUM	HIGH

Signature of the Faculty