LAB MANUAL

Subject Code : **Machine Learning Lab**
Subject Name : **6CS4-22**
Branch : Computer Science Engineering
Year : III Year/ VI Semester

Arya Group of Colleges

Department of Computer Science & Engineering

(Rajasthan Technical University, KOTA)

S.NO	CONTENTS	Page
1	Syllabus	
2	Do's and Don't's	
3	Instruction to the students	
4	Lab PEO	
5	LAB Plan	
	Experiment as per RTU syllabus	
Exp:- 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..	
	Sample Viva Question	
Exp:- 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	
	Sample Viva Question	
Exp:- 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.	
	Sample Viva Question	
Exp:- 4	Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets..	
	Sample Viva Question	
Exp:- 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.	
	Sample Viva Question	
Exp:- 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.	
	Sample Viva Question	
Exp:- 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.	
	Sample Viva Question	
Exp:- 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..	
	Sample Viva Question	
Exp:-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.	
	Sample Viva Question	
Exp:-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.	
	Sample Viva Question	

Syllabus

III Year-VI Semester: B.Tech. Computer Science and Engineering

6CS4-22: Machine Learning Lab

Credit: 1.5 **Max. Marks: 75 (IA:45, ETE:30)**

0L+0T+3P **End Term Exam: 2 Hours**

SN	List of 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 Bayesian classifier 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 aBayesian 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.

DO'S AND DONT'S

DO'S

1. Student should get the record of previous experiment checked before starting the new experiment.
2. Read the manual carefully before starting the experiment.
3. Turn off the compute before leaving the lab unless a member of lab staff has specifically told you not to do so.
4. Before switching on the power supply, get the circuit connections checked.
5. Get your outputs checked by the teacher.
6. PC and Apparatus must be handled carefully.
7. Maintain strict discipline.
8. Keep your mobile phone switched off or in vibration mode.
9. Students should get the experiment allotted for next turn, before leaving the lab.

DONT'S

1. Don't use internet, internet chat of any kind in your regular lab schedule.
2. Do not download or upload of MP3, JPG or MPEG files.
3. No games are allowed in the lab sessions.
4. No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
5. Do not leave the without permission from the teacher.
6. If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

Instructions to the Students

General Instructions

- Maintain separate observation copy for each laboratory.
- Observations or readings should be taken only in the observation copy.
- Get the readings counter signed by the faculty after the completion of the experiment.
- Maintain Index column in the observation copy and get the signature of the faculty before leaving the lab.

Before Entering the Lab

- The previous experiment should have been written in the practical file, without which the students will not be allowed to enter the lab.
- The students should have written the experiment in the observation copy that they are supposed to perform in the lab.
- The experiment written in the observation copy should have aim, apparatus required, circuit diagram/algorithm, blank observation table (if any), formula (if any), programmed (if any), model graph (if any) and space for result.

When Working in the Lab

- Necessary equipments/apparatus should be taken only from the lab assistant by making an issuing slip, which would contain name of the experiment, names of batch members and apparatus or components required.
- Never switch on the power supply before getting the permission from the faculty.

Before Leaving the Lab

- The equipments/components should be returned back to the lab assistant in good condition after the completion of the experiment.
- The students should get the signature from the faculty in the observation copy.
- They should also check whether their file is checked and counter signed in the index.

PROGRAM EDUCATION OBJECTIVES AND OUTCOMES

Lab Name: 6CS4-22: Machine Learning Lab

Class: B. Tech. III Yr VI Sem. Computer Science Engineering

L	T	P
0	0	2

External Marks: 45 **Internal Marks: 30** **Total Marks: 75**

---

(1) Program Description: To offer high quality education in the field of Computer Science Engineering and to prepare students abreast of latest global industrial and research requirements and fulfill responsibility towards community.

(2) Program Education Objectives:

List of Program Education Objectives (PEO)	
PEO-1	**Preparation:-** To prepare to pursue advanced graduate studies in computing or related disciplines and provide students broad-based education in core areas of Computer Science, including theoretical foundations, algorithms and data structures, and computer hardware, with an appropriate blend of theory and practice and to specialize in a variety of areas of Computer Science through a selection of elective courses.
PEO-2	**Core Competence:-**To provide students with a solid foundation in computer engineering field required to solve computing problems using various programming languages and Software's and students can solve problems through logical and analytical thinking.
PEO-3	**Breathe:-**To train students with good computer and engineering breadth so as to Comprehend, analyze, design, and create novel products and solutions for the real life.
PEO-4	**Professionalism:-**To inculcate in students professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, and an ability to relate Computer engineering issues to broader social context.
PEO-5	**Learning Environment:-**To provide students with an academic environment aware of excellence leadership and lifelong learning needed for successful professional career through independent studies, thesis, internships etc.

(3) Program Outcomes & it's mapping with PEO

List of Program Outcomes	
PO-1	**Engineering Knowledge**: Apply knowledge of mathematics and science, with fundamentals of Computer Science & Engineering to be able to solve complex engineering problems related to CSE.
PO-2	**Problem Analysis**: Identify, Formulate, review research literature and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO-3	**Design/Development of solutions**: Design solutions for complex engineering problems related to CSE 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.
PO-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.
PO-5	**Modern Tool Usage**: Create, Select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to computer science related complex engineering activities with an understanding of the limitations.
PO-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 CSE professional engineering practice.
PO-7	**Environment and Sustainability:** Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO-8	**Ethics**: Apply Ethical Principles and commit to professional ethics and Responsibilities and norms of the engineering practice.
PO-9	**Individual and Team Work**: Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary Settings.
PO-10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as able to comprehend and with write effective reports and design documentation, make effective presentations and give and receive clear instructions.
PO-11	**Project Management and Finance**: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
PO-12	**Life-Long Learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning the broadest context of technological change.

List of Program Specific Outcomes (PSO)	
PSO-1	**Knowledge Enhancement in Computing:** The ability to interpret the foundation and strategy of hardware and software of computer systems. Graduates can solve the problems in the areas related to algorithms, multimedia, data analytics, cloud computing, human computer interface, robotics, artificial intelligence and networking for efficient design of computer systems.
PSO-2	**Software Design and Development**: The ability to understand the software development lifecycle and methodologies of software systems. Graduate will learn competent skills and knowledge of software design process. Graduate will be acquaintance to practical proficiency with a broad area of programming concepts.

MAPPING OF PEO WITH PO & PSO

Program Education Objectives (PEO)	PROGRAM OUTCOME												PSO	
	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12	PSO-1	PSO-2
PEO-1	3	3	2	3	-	-	-	1	-	-	1	2	3	2
PEO-2	3	3	3	3	3	-	1	-	-	-	-	2	2	3
PEO-3	3	3	3	2	3	-	-	-	-	-	-	3	3	2
PEO-4	-	-	-	-	-	2	2	3	3	3	3	3	1	2
PEO-5	-	1	2	1	-	3	2	2	3	2	3	3	2	3

Note: Correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

4.Course Objectives :

The study of subject Machine Learning Lab (6CS4-22) in undergraduate program in Computer Science Engineering Branch will achieve the following major objective-

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.

5.Course Outcomes

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

6. MAPPING OF COURSE OBJECTIVE WITH PROGRAM OUTCOMES

COURSE OUTCOME		PROGRAM OUTCOME										
	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12
CO-1												
CO-2		2										
CO-3		2	2									
CO-4												

Note: Correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

7.BOOKS:-

7.1 Text books:

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems

Pattern Recognition and Machine Learning

8 Instructional Methods:

8.1 Direct Instructions:

I. White board presentation
II. Live Equipment demonstration
III. Practice on computer

8.2 Interactive Instruction:

I. Think, pair, share
II. Practical

8.3 Indirect Instructions:

I. Problem solving

1.Independent Instructions:

I. Lab Assignments,

9.Learning Materials:

1. Text/Lab Practical's/ lecturer PPT

2. Web Resources:

https://www.anaconda.com/enterprise-machine-learning-getting-started/
https://deepakdvallur.weebly.com/machine-learning-laboratory.html

10.Assessment of Outcomes:

3. Internal practical assessment (one in each semester).
4. External assessment (Conducted by RTU, Kota)

11.Outcomes will be achieved through following:

5. Lab teaching.
6. Experiment performance.
7. Video lectures.

INTRODUCTION

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 labeling.

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)

<table>
<tr><th>Supervised learning</th><th>Un Supervised learning</th><th>Instance based learning</th></tr>
<tr><td>Find-s algorithm</td><td>EM algorithm</td><td rowspan="6">Locally weighted Regression algorithm</td></tr>
<tr><td>Candidate elimination algorithm</td><td rowspan="5">K means algorithm</td></tr>
<tr><td>Decision tree algorithm</td></tr>
<tr><td>Back propagation Algorithm</td></tr>
<tr><td>Naïve Bayes Algorithm</td></tr>
<tr><td>K nearest neighbour algorithm(lazy learning algorithm)</td></tr>
</table>

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.

In clustering, 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 finds the distribution of inputs in some space. Dimensionality reduction simplifies inputs by mapping them into a lower- dimensional space. Topic modeling 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 modeling 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. Some successful applications of deep learning are computer vision and speech recognition.

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 (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

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.

Experiment – 1

1. **Object: 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.**

Program:

```
import csv

a = []

with open('enjoysport.csv', 'r') as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)

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

num_attribute = len(a[0])-1

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

for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        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("\n The hypothesis for the training instance {} is : \n"
.format(i+1),hypothesis)

print("\n The Maximally specific hypothesis for the training instance is ")
print(hypothesis)
```

Output

'Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same',True
'Sunny', 'Warm', 'High', 'Strong', 'Warm','Same',True
'Rainy', 'Cold', 'High', 'Strong', 'Warm','Change',False
'Sunny', 'Warm', 'High', 'Strong', 'Cool','Change',True

Maximally Specific set

[['Sunny', 'Warm', '?', 'Strong', '?', '?']]

Viva Question

1. What is machine learning?
2. Define supervised learning
3. Define unsupervised learning
4. Define semi supervised learning
5. Define reinforcement learning
6. What do you mean by hypotheses
7. What is classification
8. What is clustering
9. Define precision, accuracy and recall
10. Define entropy

Experiment – 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.**

PROGRAM:-

```
import numpy as np
import pandas as pd

data = pd.read_csv('enjoysport.csv')
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
    print(general_h)

    for i, h in enumerate(concepts):
        print("For Loop Starts")
        if target[i] == "yes":
            print("If 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("If 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(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
        print("\n")
        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")
```

Output

```
[('sunny', 'warm', 'normal', 'strong', 'warm','same')]
[('sunny', 'warm', 'normal', 'strong', 'warm','same')]
[('sunny', 'warm', '?', 'strong', 'warm', 'same')]
[('?', '?', '?', '?', '?', '?')]
[('sunny', '?', '?', '?', '?', '?'), ('?', 'warm', '?', '?', '?', '?'), ('?', '?', '?', '?', '?', 'same')]
[('sunny', 'warm', '?', 'strong', 'warm', 'same')]
[('sunny', 'warm', '?', 'strong', '?','?')]
[('sunny', 'warm', '?', 'strong', '?','?')]
[('sunny', '?', '?', '?', '?', '?'), ('?', 'warm', '?', '?', '?', '?
```

Viva Question

1.	Define regression
2.	How Knn is different from k-means clustering
3.	What is concept learning
4.	Define specific boundary and general boundary
5.	Define target function
6.	Define decision tree
7.	What is KNN
8.	Explain gradient descent approximation
9.	State Bayes theorem
10.	Define Bayesian belief networks

Experiment – 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.

```
PROGRAM:- import math
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"));
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset,headers

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

def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))

    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
        for y in range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1

    for x in range(len(attr)):
        dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
        pos=0
        for y in range(r):
            if data[y][col]==attr[x]:
                if delete:
                    del data[y][col]
                dic[attr[x]][pos]=data[y]
                pos+=1
    return attr,dic

def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0

    counts=[0,0]
    for i in range(2):
        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

    sums=0
    for cnt in counts:
        sums+=-1*cnt*math.log(cnt,2)
    return sums

def compute_gain(data,col):
```

```
    attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])
    for x in range(len(attr)):
        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
        total_entropy-=ratio[x]*entropies[x]
    return total_entropy

def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
        node=Node("")
        node.answer=lastcol[0]
        return node

    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
        gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]


    attr,dic=subtables(data,split,delete=True)

    for x in range(len(attr)):
        child=build_tree(dic[attr[x]],fea)
        node.children.append((attr[x],child))
    return node

def print_tree(node,level):
    if node.answer!="":
        print("  "*level,node.answer)
        return

    print("  "*level,node.attribute)
    for value,n in node.children:
        print("  "*(level+1),value)
        print_tree(n,level+2)


def classify(node,x_test,features):
    if node.answer!="":
        print(node.answer)
        return
    pos=features.index(node.attribute)
    for value, n in node.children:
        if x_test[pos]==value:
            classify(n,x_test,features)

'''Main program'''
dataset,features=load_csv("id3.csv")
node1=build_tree(dataset,features)

print("The decision tree for the dataset using ID3 algorithm is")
```

```
print_tree(node1,0)
testdata,features=load_csv("id3_test.csv")

for xtest in testdata:
    print("The test instance:",xtest)
    print("The label for test instance:",end="   ")
    classify(node1,xtest,features
```

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

Viva Question

1.	Differentiate hard and soft clustering
2.	Define variance
3.	What is inductive machine learning
4.	Why K nearest neighbour algorithm is lazy learning algorithm
5.	Why naïve Bayes is naïve
6.	Mention classification algorithms
7.	Define pruning
8.	Differentiate Clustering and classification
9.	Mention clustering algorithms
10.	Define Bias

Experiment – 4

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

PROGRAM:-

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float) # two inputs [sleep,study]
y = np.array(([92], [86], [89]), dtype=float) # one output [Expected % in Exams]
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=5000      #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)) #weight of the
link from input node to hidden node
bh=np.random.uniform(size=(1,hiddenlayer_neurons)) # bias of the link from input
node to hidden node
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons)) #weight of the
link from hidden node to output node
bout=np.random.uniform(size=(1,output_neurons)) #bias of the link from hidden node
to output node


#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)

#how much hidden layer weights contributed to error
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad
```

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

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

outut

```
Input:
[[ 0.66666667 1. ]
 [ 0.33333333 0.55555556]
 [ 1.     0.66666667]]
Actual Output:
[[ 0.92]
 [ 0.86]
 [ 0.89]]
Predicted Output:
 [[ 0.89559591]
 [ 0.88142069]
 [ 0.8928407 ]]
```

Viva Question

1.	How Are Artificial Neural Networks Different From Normal Computers?
2.	Why Use Artificial Neural Networks?
3.	What Is Simple Artificial Neuron?
4.	How Artificial Neurons Learns?
5.	List Some Commercial Practical Applications Of Artificial Neural Networks?
6.	How Artificial Neural Networks Can Be Applied In Future?
7.	What Are Cases And Variables?
8.	What Is Backprop?
9.	What Learning Rate Should Be Used For Backprop?
10.	What Are Conjugate Gradients, Levenberg-marquardt, Etc.?

Experiment – 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.

```
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

confusion matrix is as
follows [[17 0 0]
 [0 17 0]
 [0 0 11]]
Accuracy metrics

	precision	recall	f1-score	support
0	1.00	1.00	1.00	17
1	1.00	1.00	1.00	17
2	1.00	1.00	1.00	11
avg / total	1.00	1.00	1.00	45

Viva Question

1.	What is 'Overfitting' in Machine learning?
2.	What is the difference between heuristic for rule learning and heuristics for decision trees?
3.	What is Perceptron in Machine Learning?
4.	What are the two paradigms of ensemble methods?
5.	What is PAC Learning?
6.	How does a Bayesian classifier work?
7.	What is naive Bayes classifier algorithm?
8.	What are Bayesian classifiers in data mining?
9.	What is Bayes rule used for?
10.	What is the importance of Bayes Theorem?

Experiment – 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.**

```
PROGRAM:- 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

```
['about', 'am', 'amazing', 'an', 'and', 'awesome', 'beers', 'best', 'boss', 'can','deal',
'do', 'enemy', 'feel', 'fun', 'good', 'have', 'horrible', 'house', 'is', 'like', 'love','my',
'not', 'of', 'place', 'restaurant', 'sandwich', 'sick', 'stuff', 'these', 'this', 'tired', 'to',
'today', 'tomorrow', 'very', 'view', 'we', 'went', 'what', 'will', 'with', 'work']
   about am amazing an and awesome beers best boss can ... today \
0   1 0    0 0 0 0 1      0 0 0 ... 0
1   0 0    0 0 0 0 0      1 0 0 ... 0
2      0 0    1 1 0 0     0 0 0 0 ... 0
3   0 0    0 0 0 0 0      0 0 0 ... 1
4   0 0    0 0 0 0 0      0 0 0 ... 0
5   0 1            0 0 1    0 0 0 0 0 ... 0
6   0 0    0 0 0 0 0      0 0 1 ... 0
7   0 0    0 0 0 0 0      0 0 0 ... 0
8   0 1    0 0 0 0 0      0 0 0 ... 0
9   0 0    0 1 0 1 0      0 0 0 ... 0
10 0 0      0 0 0 0 0 0 0 0 ... 0
11 0 0             0 0 0    0 0 0 1 0 ... 0
12 0 0      0 1 0 1 0 0 0 0 ... 0
```

Viva Question

1.	Why is naive Bayes so 'naive' ?
2.	Explain prior probability, likelihood and marginal likelihood in context of naiveBayes algorithm?
3.	How is True Positive Rate and Recall related? Write the equation.
4.	What is the difference between covariance and correlation?
5.	What is convex hull ?
6.	What do you understand by Type I vs Type II error ?
7.	When does regularization becomes necessary in Machine Learning?
8.	What do you understand by Bias Variance trade off?
9.	What's the trade-off between bias and variance?
10.	What is the difference between supervised and unsupervised machine learning?

Experiment – 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.

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

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

print('Few examples from the dataset are given below')
print(heartDisease.head())

model =
BayesianModel([('age','trestbps'),('age','fbs'),('sex','trestbps'),('exang','trest
bps'),('trestbps','heartdisease'),('fbs','heartdisease'),('heartdisease','restecg'
),('heartdisease','thalach'),('heartdisease','chol')])

print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

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

print('\n 1. Probability of HeartDisease given Age=28')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence={'age':28})
print(q['heartdisease'])

print('\n 2. Probability of HeartDisease given cholesterol=100')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence={'chol':100})
print(q['heartdisease'])
```

Viva Question

1.	How is kNN different from k-means clustering?
2.	Explain how a ROC curve works?
3.	Define precision and recall.
4.	Explain the difference between L1 and L2 regularization?
5.	What's a Fourier transform?
6.	What is deep learning, and how does it contrast with other machine learning algorithms?
7.	What's the difference between a generative and discriminative model?
8.	What cross-validation technique would you use on a time series dataset?
9.	How is a decision tree pruned?
10.	Which is more important to you– model accuracy, or model performance?

Experiment – 8

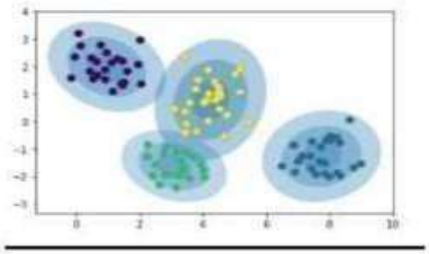
8. Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

```
PROGRAM:-
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np
#import matplotlib inline

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']

#colormap = np.array(['red', 'lime', 'black'])

# K Means Cluster
model = KMeans(n_clusters=3)
model.fit(X)
# This is what KMeans thought
model.labels_

# View the results

# Set the size of the plot
plt.figure(figsize=(14,7))

# Create a colormap
colormap = np.array(['red', 'lime', 'black'])

# Plot the Original Classifications
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')

# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')

# View the results
# Set the size of the plot
plt.figure(figsize=(14,7))
# Create a colormap
#print('The accuracy score : ',sm.accuracy_score(y, model.labels_))
#sm.confusion_matrix(y, model.labels_)

predY = np.choose(model.labels_, [0, 1, 2]).astype(np.int64)
```

```
print (predY)

#colormap = np.array(['red', 'lime', 'black'])
# Plot Orginal
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
# Plot Predicted with corrected values
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length,X.Petal_Width, c=colormap[predY], s=40)
plt.title('K Mean Classification')

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

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
#xs.sample(5)

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

y_cluster_gmm = gmm.predict(xs)
#y_cluster_gmm

plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_cluster_gmm], s=40)
plt.title('GMM Classification')

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

Output

[[1 ,0, 0, 0]
[0 ,0, 1, 0]
[1 ,0, 0, 0]
[1 ,0, 0, 0]
[1 ,0, 0, 0



Viva Question

1.	Can decision trees be used for performing clustering?
2.	For two runs of K-Mean clustering is it expected to get same clustering results?
3.	Is it possible that Assignment of observations to clusters does not change between successive iterations in K-Means
4.	Which of the following algorithm is most sensitive to outliers?
5.	What could be the possible reason(s) for producing two different dendrograms using agglomerative clustering algorithm for the same dataset?
6.	Which method is used for finding optimal of cluster in K-Mean algorithm?
7.	How is the k-nearest neighbor algorithm different from k-means clustering?
8.	Define precision and recall.
9.	What is the difference between a Generative and Discriminative Algorithm?
10.	What cross-validation technique would you use on a time series dataset?

Experiment – 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.

```
PROGRAM:- from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets

iris=datasets.load_iris()

x = iris.data
y = iris.target

print ('sepal-length', 'sepal-width', 'petal-length', 'petal-width')
print(x)
print('class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica')
print(y)

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)

#To Training the model and Nearest nighbors K=5
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)

#To make predictions on our test data
y_pred=classifier.predict(x_test)

print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

OUTPUT

Confusion matrix is as follows

[[11 0 0]

[0 9 1]

[0 1 8]]

Accuracy metrics

0 1.00 1.00 1.00 11

1 0.90 0.90 0.90 10

2 0.89 0.89 0,89 9

Avg/Total 0.93 0.93 0.93 30

Viva Question

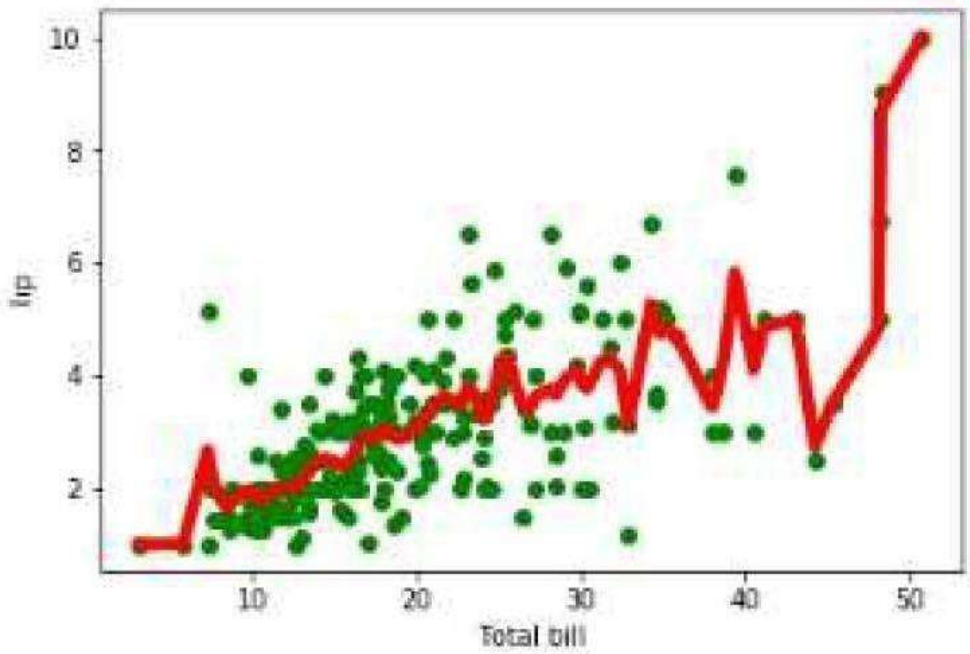
1. Difference between machine learning and deep learning
2. What is Ensemble Learning?
3. Is it feasible to compare results of datasets created over wired network and wireless ?
4. How can i access Dissolved Gas Analysis (DGA) Dataset ?
5. How to classify mixed data?
6. How to choose between classifier based on machine learning to diagnosis the fault?
7. How to make piezoelectric nano-composite (KNN=BTO)?
8. What is difference between Rand and Weights in any feature selection technique?
9. Do you think that Convolutional Neural Network is good for mobile application?
10. Can I synthesize KNN nanowires via thermal evaporation?

Experiment – 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:- import numpy as np
from bokeh.plotting import figure, show, output_notebook
from bokeh.layouts import gridplot
from bokeh.io import push_notebook

def local_regression(x0, X, Y, tau):# add bias term
 x0 = np.r_[1, x0] # Add one to avoid the loss in information
 X = np.c_[np.ones(len(X)), X]

 # fit model: normal equations with kernel
 xw = X.T * radial_kernel(x0, X, tau) # XTranspose * W

 beta = np.linalg.pinv(xw @ X) @ xw @ Y #@ Matrix Multiplication or Dot Product


 # predict value
 return x0 @ beta # @ Matrix Multiplication or Dot Product for prediction
def radial_kernel(x0, X, tau):
 return np.exp(np.sum((X - x0) ** 2, axis=1) / (-2 * tau * tau))
# Weight or Radial Kernal Bias Function

n = 1000
# generate dataset
X = np.linspace(-3, 3, num=n)
print("The Data Set ( 10 Samples) X :\n",X[1:10])
Y = np.log(np.abs(X ** 2 - 1) + .5)
print("The Fitting Curve Data Set (10 Samples) Y :\n",Y[1:10])
# jitter X
X += np.random.normal(scale=.1, size=n)
print("Normalised (10 Samples) X :\n",X[1:10])

domain = np.linspace(-3, 3, num=300)
print(" Xo Domain Space(10 Samples) :\n",domain[1:10])
def plot_lwr(tau):
 # prediction through regression
 prediction = [local_regression(x0, X, Y, tau) for x0 in domain]
 plot = figure(plot_width=400, plot_height=400)
 plot.title.text='tau=%g' % tau
 plot.scatter(X, Y, alpha=.3)
 plot.line(domain, prediction, line_width=2, color='red')
 return plot

show(gridplot([
 [plot_lwr(10.), plot_lwr(1.)],
 [plot_lwr(0.1), plot_lwr(0.01)]]))
```

Output



Viva Question

1. Python or R – Which one would you prefer for text analytics?
2. Which technique is used to predict categorical responses?
3. What is logistic regression? Or State an example when you have used logistic regression recently.
4. What are Recommender Systems?
5. Why data cleaning plays a vital role in analysis?
6. Differentiate between univariate, bivariate and multivariate analysis.
7. What do you understand by the term Normal Distribution?
8. What is Interpolation and Extrapolation?
9. What is the difference between Cluster and Systematic Sampling?
10. Are expected value and mean value different?