Dr Ambedkar Institute of Technology

Bangalore-56

Department of
Computer Science & Engineering

Laboratory Manual

For

Machine Learning Laboratory

(18CSL66)

(6th Semester B.E., CSE)

2020-21

Lab in charge:

Prof. Asha K N

Prof. Asha Rani K P

A. LABORATORYOVERVIEW

Degree:	**BE**	Programme:	**CS**
Semester:	**6**	Academic Year:	**2020-21**
Laboratory Title:	**Machine Learning Laboratory**	Laboratory Code:	**18CSL66**
L-T-P-S:	**0-0-1-0**	Duration of SEE:	**3Hrs**
CIE Marks:	**50**	SEE Marks:	**50**
Lab Manual Author:	**Asha K N** **Asha Rani K P**	Sign	

B. DESCRIPTION

1.PREREQUISITES:

- Creative thinking, sound mathematical insight and programming skills.
- Design and Analysis of Algorithms
- Python Programming

2.BASE COURSE:

- **Machine Learning(18CS62)**

3.COURSE OBJECTIVES/OUTCOMES:

Course Objectives:

This course will enable students to

1. Implement the machine learning algorithms using the Data Set.
2. Learn to use Various python tools for Machine Learning
3. Analyze and interpret the outcomes of the machine learning algorithms.

Course Outcomes:

Course Outcomes	Description	RBT Levels
The students should be able to:		
CO1	Understand and interpret the implementation procedures and python Libraries for the machine learning algorithms.	**L2**
CO2	Analyse the correctness of the datasets to apply appropriate Machine Learning algorithms.	**L3**
CO3	Design and implement Machine Learning algorithms to solve real world problems.	**L4**
CO4	Evaluate and interpret the results of the machine learning algorithms.	**L5**

4.RESOURSESREQUIRED:

- Hardware resources
  - Desktop PC
  - Windows/Linux operating system
- Software resources
  - Python
  - Anaconda IDE with Jupiter
- Datasetsfromstandardreporsitaries(Ex:https://archive.ics.uci.edu/ml/datasets.html)

Prepared by **Asha K N** **Asha Rani K P**	**Dr. Siddaraju** **HOD, CSE**

5.RELEVANCEOFTHECOURSE:

- To carry out Mini project and Main Project work

6.GENERALINSTRUCTIONS:

- Implement the program in Python IDE – Jupiter and demonstrate the same.
- External practical examination.
  - All laboratory experiments are to be included
  - Students are allowed to pick one experiment from the lot.
  - Marks distribution:Procedure+Conduction+Viva:10+30+10(50)
  - Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

7.CONTENTS:

ExptNo.	Title of the Experiments	RBT	CO
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.	L3	CO1, 2,3,4
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.	L3	CO1, 2,3,4
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.	L3	CO1, 2,3,4
4	Build an **Artificial Neural Network by implementing the Back propagation algorithm** and test the same using appropriate data sets.	L3	CO1, 2,3,4
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.	L3	CO1, 2,3,4
6	Assuming a set of documents that need to be classified, use the **naïve Bayesian Classifier model** to perform this task. Built-in API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.	L3	CO1, 2,3,4
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 Python ML library classes/API.	L3	CO1, 2,3,4
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 Python ML library classes/API in the program.	L3	CO1, 2,3,4
9	Write a program to implement **k-Nearest Neighbour algorithm** to classify the iris data set. Print both correct and wrong predictions. Python ML library classes can be used for this problem.	L3	CO1, 2,3,4
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.	L3	CO1, 2,3,4

8.REFERENCE:
1. Tom M.Mitchell, Machine Learning, India Edition2013,Mc GrawHill Education. 2. Trevor Hastie, Robert Tibshirani, Jerome Friedman, h The Elements of Statistical Learning, 2nd edition, springer series in statistics. 3. Ethem Alpaydin, Introduction to machine learning, second edition, MIT press.

C. EVALUATIONSCHEME

1. LaboratoryCIE:50Marks
   (Record writing,Viva-voce,Writing and Execution)
2. *SEE*: 50Marks

D.ARTICULATIONMATRIX

Mapping of CO to PO												
	POs											
COs	1	2	3	4	5	6	7	8	9	10	11	12
Understand and interpret the implementation procedures and python Libraries for the machine learning algorithms.	**3**	**3**	**2**		**3**							
Analyse the correctness of the data sets to apply appropriate Machine Learning algorithms.	**3**	**3**	**3**	**3**	**3**							
Design and implement Machine Learning algorithms to solve real world problems.	**3**	**3**	**3**	**3**	**3**	**2**						**2**
Evaluate and interpret the results of the machine learning algorithms.	**3**	**3**		**3**	**3**							

E.EXPERIMENTS

1.EXPERIMENT NO:1

2.TITLE:**FIND-SALGORITHM**

3. LEARNING OBJECTIVES:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:

- 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.

5. THEORY:

- The concept learning approach in machine learning, can be formulated as “Problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples”.
- Find-S algorithm for concept learning is one of the most basic algorithms of machine learning.

Find-S Algorithm

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x

    For each attribute constraint a i in h :

        If the constraint a i in h is satisfied by x then do nothing

        Else replace a i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

- It is Guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.
- Also Notice that negative examples are ignored.

Limitations of the Find-S algorithm:

- No way to determine if the only final hypothesis (found by Find-S) is consistent with data or there are more hypothesis that is consistent with data.
- Inconsistent sets of training data can mislead the finds algorithm as it ignores negative data samples.
- A good concept learning algorithm should be able to backtrack the choice of hypothesis found so that the resulting hypothesis can be improved over time. Unfortunately, Find-S provide no such method.

6. LEARNING OUTCOMES :

- Students will be able to apply Find-S algorithm to the real world problem and find the most specific hypothesis from the training data.

7. APPLICATION AREAS:

- Classification based problems.

8. PROGRAMME

```python
#read training data from csv file

import csv
with open('enjoysport.csv', 'r') as f:
    reader = csv.reader(f)
    data = list(reader)                 #convert data into list of rows

#Training data from CSV file
print("Training data\n")
for row in data:
    print(row)

attr_len=len(data[0])-1
h = ['0']*attr_len          # Initialize h to the most specific hypothesis in H

print("h=",h)
k=0

print("\nThe Hypothesis are\n")
for row in data:
    if row[-1] == 'yes':            #For each positive training instance x
        j = 0
        for col in row:             #For each attribute constraint a, in h
            if col != 'yes':        #replace a, in h by the next more general constraint
                if col != h[j] and h[j] == '0':
                    h[j] = col
                elif col != h[j] and h[j] != '0':
                    h[j] = '?'

            j = j + 1

    print("h",k,"=",h) #print all Hypothesis
    k=k+1
print('\nMaximally Specific Hypothesis: \n',"h",k-1,"=", h)      #print final hypothes
```

9. OUTPUT

```
Training data

['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']
h= ['0', '0', '0', '0', '0', '0']

The Hypothesis are

h 0 = ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
h 1 = ['sunny', 'warm', '?', 'strong', 'warm', 'same']
h 2 = ['sunny', 'warm', '?', 'strong', 'warm', 'same']
h 3 = ['sunny', 'warm', '?', 'strong', '?', '?']

Maximally Specific Hypothesis:
 h 3 = ['sunny', 'warm', '?', 'strong', '?', '?']
```

1. EXPERIMENT NO: 2

2. TITLE: **CANDIDATE-ELIMINATION ALGORITHM**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:
- 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.

5. THEORY:
- The key idea in the Candidate-Elimination algorithm is to output a description of the set of all hypotheses consistent with the training examples.
- It computes the description of this set without explicitly enumerating all of its members.
- This is accomplished by using the more-general-than partial ordering and maintaining a compact representation of the set of consistent hypotheses.
- The algorithm represents the set of all hypotheses consistent with the observed training examples. This subset of all hypotheses is called the version space with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.
- A version space can be represented with its general and specific boundary sets.
- The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S.
- Given only these two sets S and G, it is possible to enumerate all members of a version space by generating hypotheses that lie between these two sets in general-to-specific partial ordering over hypotheses. Every member of the version space lies between these boundaries

Algorithm

1. Initialize G to the set of maximally general hypotheses in H
2. Initialize S to the set of maximally specific hypotheses in H
3. For each training example d, do

If d is a positive example

Remove from G any hypothesis inconsistent with d , For each hypothesis s in S that is not consistent with d , Remove s from S

Add to S all minimal generalizations h of s such that h is consistent with d, and some member of G is more general than h

Remove from S, hypothesis that is more general than another hypothesis in S

If d is a negative example

Remove from S any hypothesis inconsistent with d For each hypothesis g in G that is not consistent with d Remove g from G

Add to G all minimal specializations h of g such that h is consistent with d, and some member of S is more specific than h

Remove from G any hypothesis that is less general than another hypothesis in G

6. LEARNING OUTCOMES :
- The students will be able to apply candidate elimination algorithm and output a description of the set of all hypotheses consistent with the training examples

7. APPLICATION AREAS:
- Classification based problems.

8. PROGRAMME

```python
import csv

with open('enjoysport.csv', 'r') as f:
    reader = csv.reader(f)
    data = list(reader)

#Training data from CSV file
print("Training data\n")
for row in data:
    print(row)
print("-------------------------------------")

attr_len=len(data[0])-1

#initialize Specific and General Hypothesis
S = ['0']*attr_len
G = ['?']*attr_len
temp=[]             # altered G

print("The Hypothesis are\n")
print("S=",S)
print("G=",G)
print("-------------------------------------")
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('---------------------------------')
```

Output:

```
Training data

['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 Hypothesis are

S= ['0', '0', '0', '0', '0', '0']
G= ['?', '?', '?', '?', '?', '?']
--------------------------------------
S= ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
G= ['?', '?', '?', '?', '?', '?']
-----------------------------------
S= ['sunny', 'warm', '?', 'strong', 'warm', 'same']
G= ['?', '?', '?', '?', '?', '?']
-----------------------------------
S= ['sunny', 'warm', '?', 'strong', 'warm', 'same']
G= [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
-----------------------------------
S= ['sunny', 'warm', '?', 'strong', '?', '?']
G= [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
-----------------------------------
```

1. EXPERIMENT NO: 3

2. TITLE: **ID3 ALGORITHM**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:
- 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 anew sample.

5. THEORY:
- ID3 algorithm is a basic algorithm that learns decision trees by constructing them topdown,beginning with the question "which attribute should be tested at the root of the tree?".
- To answer this question, each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples. The best attribute is selectedand used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node (i.e., down the branch corresponding to the example's value for this attribute).
- The entire process is then repeated using the training examples associated with eachdescendant node to select the best attribute to test at that point in the tree.
- A simplified version of the algorithm, specialized to learning boolean-valued functions (i.e.,concept learning), is described below.

ID3(*Examples*, *Target_attribute*, *Attributes*)

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

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - *A* ← the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* ← *A*
  - For each possible value, v_i, of *A*,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for *A*
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {*A*}))
- End
- Return *Root*

---

6. LEARNING OUTCOMES :

- The student will be able to demonstrate the working of the decision tree based ID3algorithm, use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

7. APPLICATION AREAS:

- Classification related problem areas

8. PROGRAMME:

```
import pandas as pd
import math

df = pd.read_csv('id3.csv')
print("\n Input Data Set is:\n", df)

t = df.keys()[-1]
print('Target Attribute is: ', t)
attribute_names = list(df.keys())
attribute_names.remove(t)
print('Predicting Attributes: ', attribute_names)


def entropy(probs):
    return sum( [-prob*math.log(prob, 2) for prob in probs])


def entropy_of_list(ls,value):
    from collections import Counter
    cnt = Counter(x for x in ls)
    print('Target attribute class count(Yes/No)=',dict(cnt))
    total_instances = len(ls)
    print("Total no of instances/records associated with {0} is: {1}".format(value,total_instances ))
    probs = [x / total_instances for x in cnt.values()]
    return entropy(probs)

def information_gain(df, split_attribute, target_attribute,battr):
    print("\n\n-----Information Gain Calculation of ",split_attribute, " --------")
    df_split = df.groupby(split_attribute)
    glist=[]
    for gname,group in df_split:
        print('Grouped Attribute Values \n',group)
        glist.append(gname)
    nobs = len(df.index)
    df_agg1=df_split.agg({target_attribute:lambda x:entropy_of_list(x, glist.pop())})
    df_agg2=df_split.agg({target_attribute :lambda x:len(x)/nobs})
    df_agg1.columns=['Entropy']
    df_agg2.columns=['Proportion']
    new_entropy = sum( df_agg1['Entropy'] * df_agg2['Proportion'])
    if battr !='S':
        old_entropy = entropy_of_list(df[target_attribute],'S-'+df.iloc[0][df.columns.get_loc(battr)])
    else:
        old_entropy = entropy_of_list(df[target_attribute],battr)
    return old_entropy - new_entropy


```

```python
def id3(df, target_attribute, attribute_names, default_class=None,default_attr='S'):
    from collections import Counter
    cnt = Counter(x for x in df[target_attribute])     # class of YES /NO
    if len(cnt) == 1:
        return next(iter(cnt))
    elif df.empty or (not attribute_names):
        return default_class
    else:
        default_class = max(cnt.keys())
        gainz=[]
        for attr in attribute_names:
            ig= information_gain(df, attr, target_attribute,default_attr)
            gainz.append(ig)
            print('Information gain of ',attr,' is : ',ig)
        index_of_max = gainz.index(max(gainz))
        best_attr = attribute_names[index_of_max]
        print("\nAttribute with the maximum gain is: ", best_attr)
        tree = {best_attr:{}}
        remaining_attribute_names =[i for i in attribute_names if i != best_attr]
        for attr_val, data_subset in df.groupby(best_attr):
            subtree = id3(data_subset,target_attribute, remaining_attribute_names,default_class,best_attr)
            tree[best_attr][attr_val] = subtree
        return tree


from pprint import pprint

tree = id3(df,t,attribute_names)
print("\nThe Resultant Decision Tree is:")
pprint(tree)
```

Output:

Input Data Set is:

	Outlook	Temperature	Humidity	Wind	Answer
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes
5	rain	cool	normal	strong	no
6	overcast	cool	normal	strong	yes
7	sunny	mild	high	weak	no
8	sunny	cool	normal	weak	yes
9	rain	mild	normal	weak	yes
10	sunny	mild	normal	strong	yes
11	overcast	mild	high	strong	yes
12	overcast	hot	normal	weak	yes
13	rain	mild	high	strong	no

```
Target Attribute is:  Answer
Predicting Attributes:  ['Outlook', 'Temperature', 'Humidity', 'Wind']
```

```
-----Information Gain Calculation of  Outlook  --------
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
2   overcast         hot     high    weak    yes
6   overcast        cool   normal  strong    yes
11  overcast        mild     high  strong    yes
12  overcast         hot   normal    weak    yes
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
3     rain        mild     high    weak    yes
4     rain        cool   normal    weak    yes
5     rain        cool   normal  strong     no
9     rain        mild   normal    weak    yes
13    rain        mild     high  strong     no
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
0    sunny         hot     high    weak     no
1    sunny         hot     high  strong     no
7    sunny        mild     high    weak     no
8    sunny        cool   normal    weak    yes
10   sunny        mild   normal  strong    yes
Target attribute class count(Yes/No)= {'yes': 4}
Total no of instances/records associated with sunny is: 4
Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}
Total no of instances/records associated with rain is: 5
Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}
Total no of instances/records associated with overcast is: 5
Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}
Total no of instances/records associated with S is: 14
Information gain of  Outlook  is :  0.2467498197744391
```

```
-----Information Gain Calculation of  Temperature  --------
Grouped Attribute Values
     Outlook Temperature Humidity    Wind Answer
4      rain        cool   normal    weak    yes
5      rain        cool   normal  strong     no
6  overcast        cool   normal  strong    yes
8     sunny        cool   normal    weak    yes
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
0      sunny         hot     high    weak     no
1      sunny         hot     high  strong     no
2   overcast         hot     high    weak    yes
12  overcast         hot   normal    weak    yes
```

```
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
3        rain        mild     high    weak    yes
7       sunny        mild     high    weak     no
9        rain        mild   normal    weak    yes
10      sunny        mild   normal  strong    yes
11   overcast        mild     high  strong    yes
13       rain        mild     high  strong     no
Target attribute class count(Yes/No)= {'yes': 3, 'no': 1}
Total no of instances/records associated with mild is: 4
Target attribute class count(Yes/No)= {'no': 2, 'yes': 2}
Total no of instances/records associated with hot is: 4
Target attribute class count(Yes/No)= {'yes': 4, 'no': 2}
Total no of instances/records associated with cool is: 6
Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}
Total no of instances/records associated with S is: 14
Information gain of  Temperature  is :  0.029222565658954647


-----Information Gain Calculation of  Humidity  --------
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
0       sunny         hot     high    weak     no
1       sunny         hot     high  strong     no
2    overcast         hot     high    weak    yes
3        rain        mild     high    weak    yes
7       sunny        mild     high    weak     no
11   overcast        mild     high  strong    yes
13       rain        mild     high  strong     no
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
4        rain        cool   normal    weak    yes
5        rain        cool   normal  strong     no
6    overcast        cool   normal  strong    yes
8       sunny        cool   normal    weak    yes
9        rain        mild   normal    weak    yes
10      sunny        mild   normal  strong    yes
12   overcast         hot   normal    weak    yes
Target attribute class count(Yes/No)= {'no': 4, 'yes': 3}
Total no of instances/records associated with normal is: 7
Target attribute class count(Yes/No)= {'yes': 6, 'no': 1}
Total no of instances/records associated with high is: 7
Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}
Total no of instances/records associated with S is: 14
Information gain of  Humidity  is :  0.15183550136234136
```

```
-----Information Gain Calculation of  Wind  --------
Grouped Attribute Values
      Outlook Temperature Humidity    Wind Answer
1       sunny         hot     high  strong     no
5        rain        cool   normal  strong     no
6    overcast        cool   normal  strong    yes
10      sunny        mild   normal  strong    yes
11   overcast        mild     high  strong    yes
13       rain        mild     high  strong     no
Grouped Attribute Values
      Outlook Temperature Humidity  Wind Answer
0       sunny         hot     high  weak     no
2    overcast         hot     high  weak    yes
3        rain        mild     high  weak    yes
4        rain        cool   normal  weak    yes
7       sunny        mild     high  weak     no
8       sunny        cool   normal  weak    yes
9        rain        mild   normal  weak    yes
12   overcast         hot   normal  weak    yes
Target attribute class count(Yes/No)= {'no': 3, 'yes': 3}
Total no of instances/records associated with weak is: 6
Target attribute class count(Yes/No)= {'no': 2, 'yes': 6}
Total no of instances/records associated with strong is: 8
Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}
Total no of instances/records associated with S is: 14
Information gain of  Wind  is :  0.04812703040826927

Attribute with the maximum gain is:  Outlook



-----Information Gain Calculation of  Temperature  --------
Grouped Attribute Values
   Outlook Temperature Humidity    Wind Answer
4    rain        cool   normal    weak    yes
5    rain        cool   normal  strong     no
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
3     rain        mild     high    weak    yes
9     rain        mild   normal    weak    yes
13    rain        mild     high  strong     no
Target attribute class count(Yes/No)= {'yes': 1, 'no': 1}
Total no of instances/records associated with mild is: 2
Target attribute class count(Yes/No)= {'yes': 2, 'no': 1}
Total no of instances/records associated with cool is: 3
Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}
Total no of instances/records associated with S-rain is: 5
Information gain of  Temperature  is :  0.01997309402197489
```

```
-----Information Gain Calculation of  Humidity  --------
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
3     rain        mild     high    weak    yes
13    rain        mild     high  strong     no
Grouped Attribute Values
   Outlook Temperature Humidity    Wind Answer
4    rain        cool   normal    weak    yes
5    rain        cool   normal  strong     no
9    rain        mild   normal    weak    yes
Target attribute class count(Yes/No)= {'yes': 1, 'no': 1}
Total no of instances/records associated with normal is: 2
Target attribute class count(Yes/No)= {'yes': 2, 'no': 1}
Total no of instances/records associated with high is: 3
Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}
Total no of instances/records associated with S-rain is: 5
Information gain of  Humidity  is :  0.01997309402197489


-----Information Gain Calculation of  Wind  --------
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
5     rain        cool   normal  strong     no
13    rain        mild     high  strong     no
Grouped Attribute Values
   Outlook Temperature Humidity  Wind Answer
3    rain        mild     high  weak    yes
4    rain        cool   normal  weak    yes
9    rain        mild   normal  weak    yes
Target attribute class count(Yes/No)= {'no': 2}
Total no of instances/records associated with weak is: 2
Target attribute class count(Yes/No)= {'yes': 3}
Total no of instances/records associated with strong is: 3
Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}
Total no of instances/records associated with S-rain is: 5
Information gain of  Wind  is :  0.9709505944546686

Attribute with the maximum gain is:  Wind

-----Information Gain Calculation of  Temperature  --------
Grouped Attribute Values
   Outlook Temperature Humidity  Wind Answer
8    sunny        cool   normal  weak    yes
Grouped Attribute Values
```

```
   Outlook Temperature Humidity    Wind Answer
0   sunny         hot      high    weak     no
1   sunny         hot      high  strong     no
Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
7    sunny        mild      high    weak     no
10   sunny        mild    normal  strong    yes
Target attribute class count(Yes/No)= {'yes': 1}
Total no of instances/records associated with mild is: 1
Target attribute class count(Yes/No)= {'no': 2}
Total no of instances/records associated with hot is: 2
Target attribute class count(Yes/No)= {'no': 1, 'yes': 1}
Total no of instances/records associated with cool is: 2
Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}
Total no of instances/records associated with S-sunny is: 5
Information gain of  Temperature  is :  0.5709505944546686


 -----Information Gain Calculation of  Humidity  --------
 Grouped Attribute Values
    Outlook Temperature Humidity    Wind Answer
 0   sunny         hot      high    weak     no
 1   sunny         hot      high  strong     no
 7   sunny        mild      high    weak     no
 Grouped Attribute Values
     Outlook Temperature Humidity    Wind Answer
 8    sunny        cool    normal    weak    yes
 10   sunny        mild    normal  strong    yes
 Target attribute class count(Yes/No)= {'no': 3}
 Total no of instances/records associated with normal is: 3
 Target attribute class count(Yes/No)= {'yes': 2}
 Total no of instances/records associated with high is: 2
 Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}
 Total no of instances/records associated with S-sunny is: 5
 Information gain of  Humidity  is :  0.9709505944546686

   -----Information Gain Calculation of  Wind  --------
   Grouped Attribute Values
       Outlook Temperature Humidity    Wind Answer
   1     sunny         hot      high  strong     no
   10    sunny        mild    normal  strong    yes
   Grouped Attribute Values
      Outlook Temperature Humidity  Wind Answer
   0    sunny         hot      high  weak     no
   7    sunny        mild      high  weak     no
   8    sunny        cool    normal  weak    yes
```

```
Target attribute class count(Yes/No)= {'no': 1, 'yes': 1}
Total no of instances/records associated with weak is: 2
Target attribute class count(Yes/No)= {'no': 2, 'yes': 1}
Total no of instances/records associated with strong is: 3
Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}
Total no of instances/records associated with S-sunny is: 5
Information gain of  Wind  is :  0.01997309402197489

Attribute with the maximum gain is:  Humidity

The Resultant Decision Tree is:
{'Outlook': {'overcast': 'yes',
             'rain': {'Wind': {'strong': 'no', 'weak': 'yes'}},
             'sunny': {'Humidity': {'high': 'no', 'normal': 'yes'}}}}
```

1. EXPERIMENT NO: 4

2. TITLE: **BACKPROPAGATION ALGORITHM**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:
- Build an Artificial Neural Network by implementing the Backpropagation algorithm and testthe same using appropriate data sets.

5. THEORY:
- Artificial neural networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from examples.
- Algorithms such as BACKPROPAGATION gradient descent to tune network parameters tobest fit a training set of input-output pairs.
- ANN learning is robust to errors in the training data and has been successfully applied to problems such as interpreting visual scenes, speech recognition, and learning robot controlstrategies.

Backpropagation algorithm:

1. Input the instance x, to the network and compute the output o_u of every unit u in the network.
   Propagate the errors backward through the network
2. For each network unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each network unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

6. LEARNING OUTCOMES :
- The student will be able to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

7. APPLICATION AREAS:
- Speech recognition, Character recognition, Human Face recognition

8. PROGRAMME:

```python
import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0) #maximum of X array longitudinally
y = y/100

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

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

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

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

#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):
    #Forward Propogation
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp= outinp1+bout
    output = sigmoid(outinp)

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

    wout += hlayer_act.T.dot(d_output) *lr   # dotproduct of nextlayererror and currentlayerop
    bout += np.sum(d_output, axis=0,keepdims=True) *lr
    wh += X.T.dot(d_hiddenlayer) *lr
    bh += np.sum(d_hiddenlayer, axis=0,keepdims=True) *lr
print("Input: \n" ,X)
print("Actual Output: \n" ,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:

 [[0.90659823]
 [0.89630074]
 [0.90144612]]
```

1. EXPERIMENT NO: 5

2. TITLE: **NAÏVE BAYESIAN CLASSIFIER**

3. LEARNING OBJECTIVES:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:

- 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.

5. THEORY:

Naive Bayes algorithm: Naive Bayes algorithm is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naïve Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c).Look at the equation below:

$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c\mid \mathbf{X}) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

where

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).P(c) is the prior probability of class.

P(x|c) is the likelihood which is the probability of predictor given class.P(x) is the prior probability of predictor.

The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f (x) can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values (a1, a2, ... ,an). The learner is asked to predict the target value, or classification, for this new instance.The Bayesian approach to classifying the new instance is to assign the most probable target value,

Algorithm

Algorithm:
NaiveBaiseClassifier(training_examples, New_Instance)

Each instance ***x*** is described by a conjunction of attribute values(a_i) and the target V can take j finite set of values.

a. For each value j in target estimate the P(Vj)
b. For each attribute in the training example estimate Estimate the P(ai|Vj)
c. Classify each instance as per the rule in equation

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

Where V_{NB} denotes the target value output by the naive Bayes classifier

d. Output V_{NB}

6. LEARNING OUTCOMES :

- The student will be able to apply naive Bayesian classifier for the relevant problem and analyse the results.

7. APPLICATION AREAS:

- Real time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, itcould be used for making predictions in real time.
- Multi class Prediction: This algorithm is also well known for multi class prediction feature.Here we can predict the probability of multiple classes of target variable.
- Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly usedin text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- Recommendation System: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques tofilter unseen information and predict whether a user would like a given resource or not

8. PROGRAMME

```python
import numpy as np
import math
import csv

def read_data(filename):
    with open(filename,"r") as csvfile:
        datareader=csv.reader(csvfile)
        traindata=list(datareader)
    metadata=traindata[0]    #attributes name
    traindata=traindata[1:]  #training examples
    return (metadata, traindata)

def splitdataset(dataset, splitratio):
    trainsize=int(len(dataset)*splitratio)
    trainset=[]
    testset=list(dataset)
    i=0
    while len(trainset)<trainsize:
        trainset.append(testset.pop(i))
    return [trainset,testset]

def classify(data,test):
    totalsize=data.shape[0]
    print("\n")
    print("training data size=",totalsize)
    print("test data size=",test.shape[0])

    countyes=0
    countno=0
    probyes=0
    probno=0
    print("\n")
    print("target \t count \t probolaity")
    for x in range(data.shape[0]):
        if data[x,data.shape[1]-1] =='yes':
            countyes+=1
        if data[x,data.shape[1]-1] =='no':
            countno+=1
    probYes=countyes/totalsize
    probNo=countno/totalsize

    print("Yes \t", countyes,"\t", probYes)
    print("No \t", countno,"\t", probNo)

    prob0=np.zeros((test.shape[1]-1))
    prob1=np.zeros((test.shape[1]-1))

    accuracy=0
    print("\n")
    print("instance \t prediction \t target")

    for t in range(test.shape[0]):
        for k in range (test.shape[1]-1):
            count1=count0=0
            for j in range(data.shape[0]):
                #how many times appeared with no
                if test[t,k]==data[j,k] and data[j,data.shape[1]-1]=='no':
                    count0+=1
                #how many times appeared with yes
```

```python
                if test[t,k]==data[j,k] and data[j,data.shape[1]-1]=='yes':
                    count1+=1
            prob0[k]=count0/countno
            prob1[k]=count1/countyes
        probno=probNo
        probyes=probYes
        for i in range(test.shape[1]-1):
            probno=probno*prob0[i]
            probyes=probyes*prob1[i]
        if probno>probyes:
            predict='no'
        else:
            predict='yes'

        print("  ",t+1," \t\t",predict," \t\t",test[t,test.shape[1]-1])
        if predict==test[t,test.shape[1]-1]:
            accuracy+=1
    finalaccuracy=(accuracy/test.shape[0])*100
    print("\n Accuracy=",finalaccuracy,"%")



metadata,traindata=read_data("enjoysport.csv")
print("\n The attribute names of training data are:",metadata)
splitratio=0.7
trainset, testset=splitdataset(traindata, splitratio)
training=np.array(trainset)

print("\n The Training data set are:")
for x in training:
    print(x)

testing=np.array(testset)
print("\n The Test data set are:")
for x in testing:
    print(x)

classify(training, testing)
```

Output:

```
The attribute names of training data are: ['Outlook', 'Temperature', 'Humidity', 'Wind', 'Answer']

 The Training data set are:
['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']

 The Test data set are:
['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']
```

```
training data size= 9
test data size= 5
instance          prediction        target
   1              yes               yes
   2              no                yes
   3              yes               yes
   4              yes               yes
   5              no                no
Accuracy= 80.0 %

 target    count    probolaity
 Yes       5        0.5555555555555556
 No        4        0.4444444444444444
```

1. EXPERIMENT NO: 6

2. TITLE: **DOCUMENT CLASSIFICATION USING NAÏVE BAYESIAN CLASSIFIER**

3. LEARNING OBJECTIVES:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:

- Assuming a set of documents that need to be classified, use the naïve Bayesian Classifiermodel 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.

5. THEORY:

Naïve Bayes classifier algorithm for text classification

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in Examples*
   - *Vocabulary* ← the set of all distinct words and other tokens occurring in any text document from *Examples*
2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*
   - For each target value v_j in V do
     - $docs_j$ ← the subset of documents from *Examples* for which the target value is v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j$ ← a single document created by concatenating all members of $docs_j$
     - n ← total number of distinct word positions in $Text_j$
     - for each word w_k in *Vocabulary*
       - n_k ← number of times word w_k occurs in $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

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

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

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

Confusion Matrix

A confusion matrix, also called a contingency table or error matrix, is used to visualize the performance of a classifier.

Confusion Matrix		Predicted classes	
		negative	positive
Actual classes	negative	TN	FP
	positive	FN	TP

- **Accuracy**

Accuracy is a statistical measure which is defined as the quotient of correct predictions (both True positives (TP) and True negatives (TN)) made by a classifier divided by the sum of all predictions made by the classifier, including False positives (FP) and False negatives (FN).

The formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** Precision is the ratio of the correctly identified positive cases to all the predicted positive cases, i.e. the correctly and the incorrectly predicted cases as positive. Precision is the fraction of retrieved documents that are relevant to the query.
- The formula:

$$precision = \frac{TP}{TP + FP}$$

- **Recall**, also known as sensitivity, is the ratio of the correctly identified positive cases to all the actual positive cases, which is the sum of the "False Negatives" and "True Positives".
- The formula:

$$recall = \frac{TP}{TP + FN}$$

6. LEARNINGOUTCOMES:

- The student will be able to apply naive bayesian classifier for document classification and analyse the results.

7. APPLICATIONAREAS:

- Applicable in document classification

8. PROGRAMME

```python
import pandas as pd
msg=pd.read_csv('D:\\data6.csv',names=['message','label']) #Tabular form data
print('Total instances in the dataset:',msg.shape[0])

msg['labelnum']=msg.label.map({'pos':1,'neg':0})
#print(msg)
X=msg.message
Y=msg.labelnum

#print(X)
#print(Y)


# 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('\nDataset is split into Training and Testing samples')
print('Total training instances :', ytrain.shape[0])
print('Total testing instances :', ytest.shape[0])

# Output of count vectoriser is a sparse matrix
# CountVectorizer - stands for 'feature extraction'
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain) #Sparse matrix
xtest_dtm = count_vect.transform(xtest)
print('\nTotal features extracted using CountVectorizer:',xtrain_dtm.shape[1])

print('\nThe words or Tokens in the text documents\n')
print(count_vect.get_feature_names())


# 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('\nAccuracy metrics')
print('==================')
print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))

print('Recall :',metrics.recall_score(ytest,predicted), '\nPrecison :',metrics.precision_score(ytest,predicted))
print('Confusion matrix')
print('==================')
print(metrics.confusion_matrix(ytest,predicted))
```

Output:

```
Total instances in the dataset: (18, 2)

Dataset is split into Training and Testing samples
Total training instances : 13
Total testing instances : 5

Total features extracted using CountVectorizer: 42

The words or Tokens in the text documents

['am', 'amazing', 'an', 'awesome', 'bad', 'best', 'boss', 'can', 'dance', 'deal', 'do', 'fun', 'good', 'great', 'have', 'holida
y', 'horrible', 'is', 'juice', 'like', 'locality', 'love', 'my', 'not', 'of', 'place', 'restaurant', 'sandwich', 'stay', 'stuf
f', 'taste', 'that', 'the', 'this', 'tired', 'to', 'tomorrow', 'we', 'what', 'will', 'with', 'work']

Accuracy metrics
==================
Accuracy of the classifer is 0.6
Recall : 1.0
Precison : 0.5
Confusion matrix
==================
[[1 2]
 [0 2]]
```

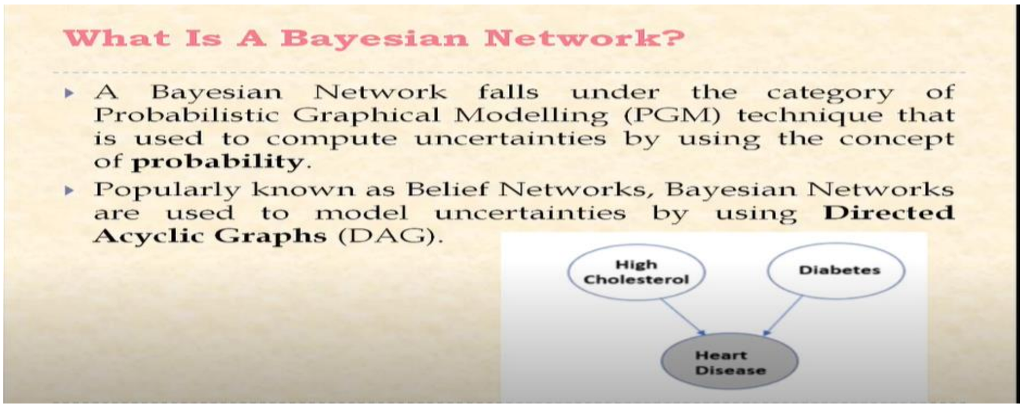
1. EXPERIMENT NO: 7

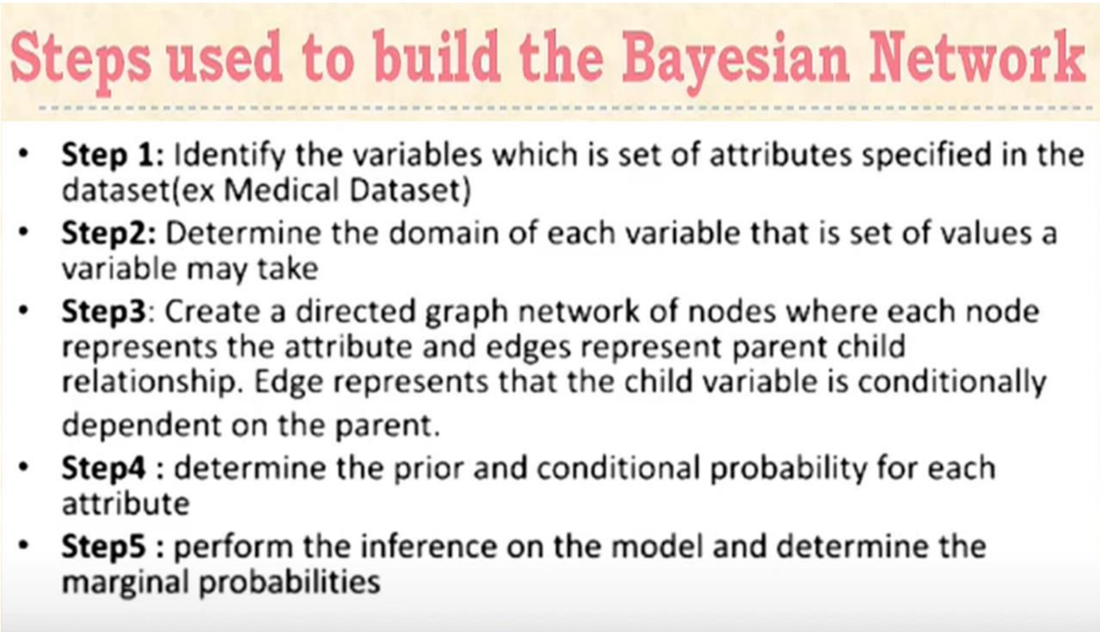
2. TITLE: **BAYESIAN NETWORK**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:
- 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.

5. THEORY:





6. LEARNING OUTCOMES:
- The student will be able to apply bayesian network for the medical data and demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

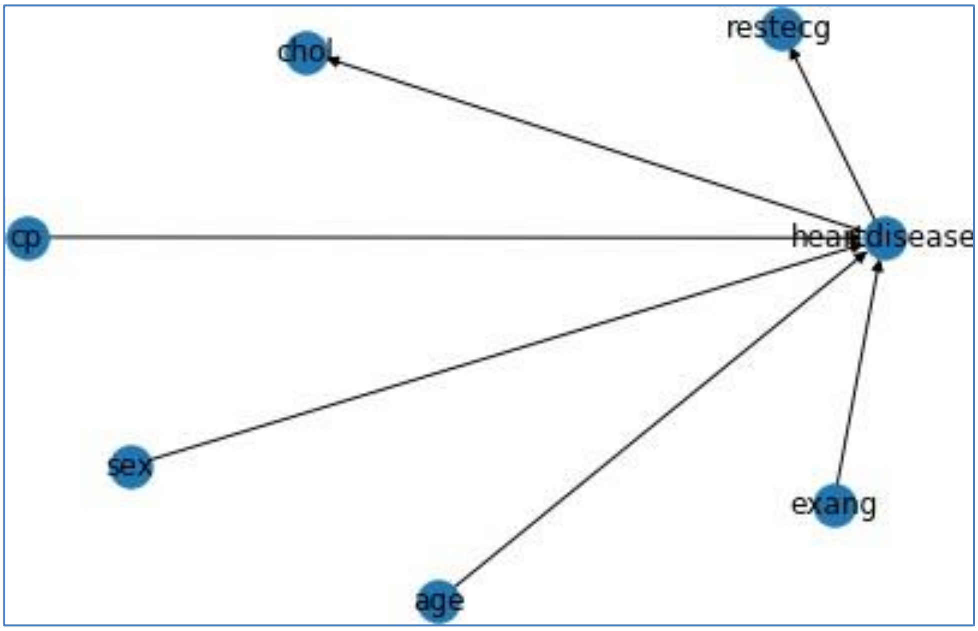
7. APPLICATION AREAS:
- Applicable in prediction and classification
- Gene Regulatory Networks
- Medicine
- Biomonitoring
- Document Classification
- Information Retrieval
- Semantic Search

8. PROGRAMME

```
import pandas as pd
import numpy as np

#read Cleveland Heart Disease data
data = pd.read_csv('heart.csv')
data = data.replace('?',np.nan)
#display the data
print('Sample instances from the dataset are given below')
print(data.head())
#display the Attributes names and datatyes
print('\n Attributes and datatypes')
print(data.dtypes)

from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

#Creat Model- Bayesian Network
#Defining the model structure. We can define the network by just -
#passing a list of edges.
model =BayesianModel([('age','heartdisease'),('sex','heartdisease'),
                      ('exang','heartdisease'),('cp','heartdisease'),
                      ('heartdisease','restecg'),('heartdisease','chol')])
import networkx as nx
import matplotlib.pyplot as plt
nx.draw(model, with_labels = True);
plt.show()

#Learning CPDs using Maximum Likelihood Estimators for all the variables
print('\n Learning CPD using Maximum likelihood estimators')
model.fit(data,estimator=MaximumLikelihoodEstimator)

#print(model.get_cpds('cp'))

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

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

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

Output:

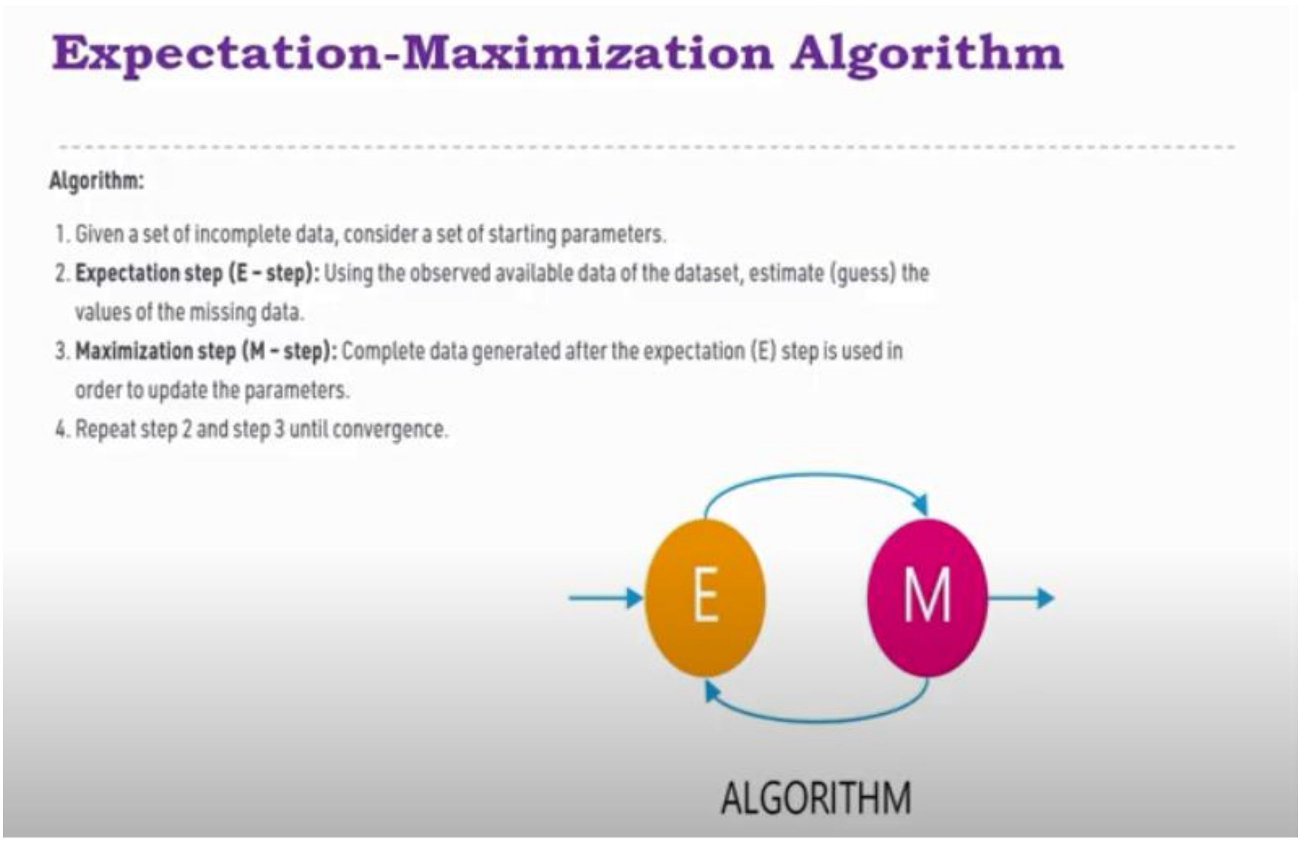
```
Sample instances from the dataset are given below
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   1       145   233    1        2      150      0      2.3      3
1   67    1   4       160   286    0        2      108      1      1.5      2
2   67    1   4       120   229    0        2      129      1      2.6      2
3   37    1   3       130   250    0        0      187      0      3.5      3
4   41    0   2       130   204    0        2      172      0      1.4      1

  ca thal  heartdisease
0  0    6             0
1  3    3             2
2  2    7             1
3  0    3             0
4  0    3             0

 Attributes and datatypes
age               int64
sex               int64
cp                int64
trestbps          int64
chol              int64
fbs               int64
restecg           int64
thalach           int64
exang             int64
oldpeak         float64
slope             int64
ca               object
thal             object
heartdisease      int64
dtype: object
```



```
Learning CPD using Maximum likelihood estimators

Finding Elimination Order: : 100%|██████████████████████████████| 5/5 [00:00<00:00, 1666.79it/s]
Eliminating: cp: 100%|██████████████████████████████████████| 5/5 [00:00<00:00, 277.74it/s]


Inferencing with Bayesian Network:

1.Probability of HeartDisease given evidence=restecg :1
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.1012 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.2392 |
+-----------------+---------------------+
| heartdisease(3) |              0.2015 |
+-----------------+---------------------+
| heartdisease(4) |              0.4581 |
+-----------------+---------------------+
```

```
2.Probability of HeartDisease given evidence= cp:2


Finding Elimination Order: : 100%|██████████████████████████████| 5/5 [00:00<00:00, 4983.73it/s]
Eliminating: chol: 100%|████████████████████████████████████| 5/5 [00:00<00:00, 454.64it/s]

+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

1. EXPERIMENT NO: 8
2. TITLE: **CLUSTERING BASED ON EM ALGORITHM AND K-MEANS**
3. LEARNING OBJECTIVES:
   - Make use of Data sets in implementing the machine learning algorithms.
   - Implement ML concepts and algorithms in Python
4. AIM:
   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.
5. THEORY:



K-Means Algorithm

- **K-Means** algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**.
- It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.
- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

6. LEARNINGOUTCOMES:

- The students will be capable to apply EM algorithm and k-Means algorithm for clustering and analyse the results.

7. APPLICATIONAREAS:

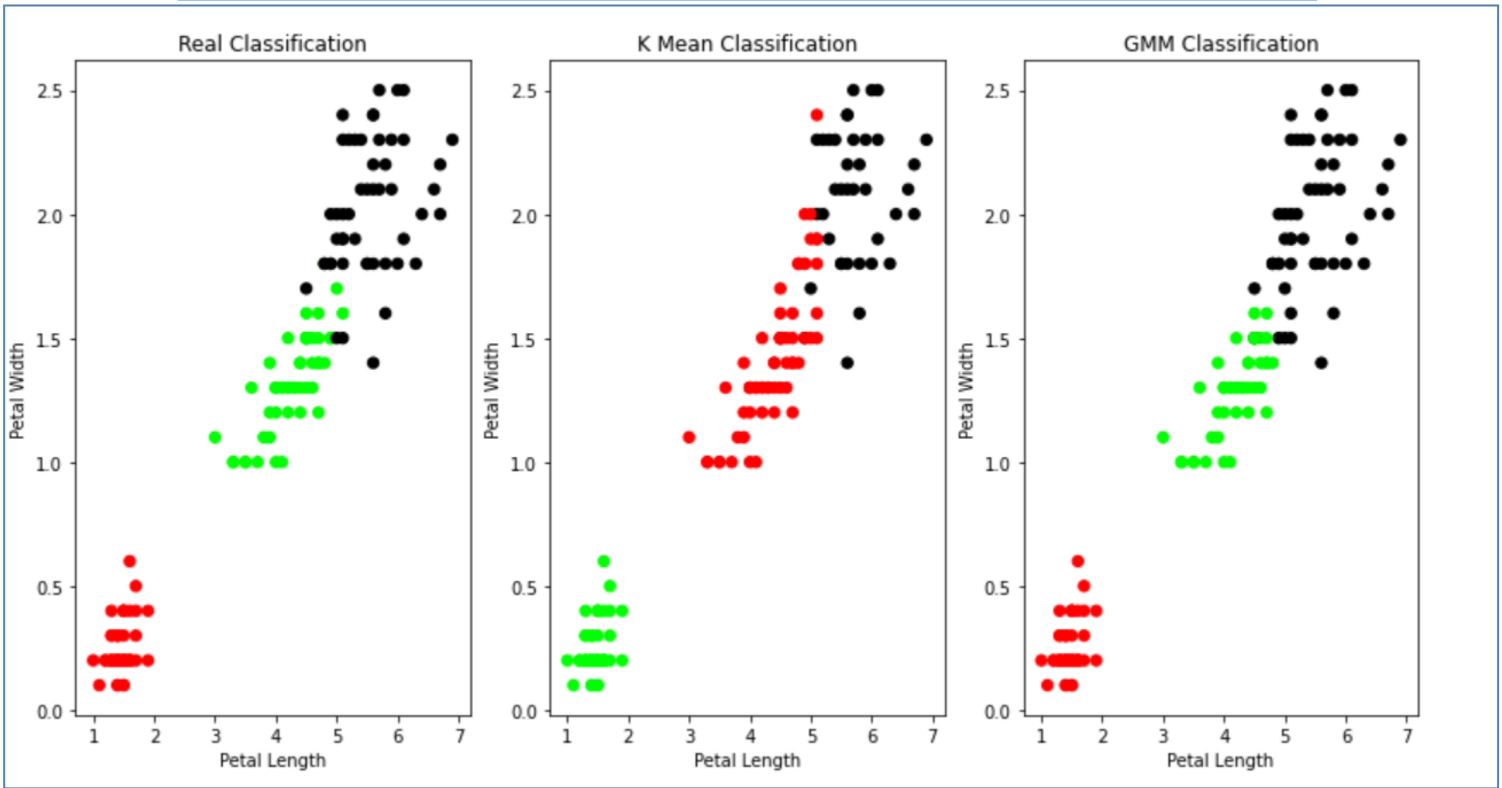
- Text mining
- Pattern recognition
- Image analysis

8. PROGRAMME:

```python
import matplotlib.pyplot as plt
from sklearn import datasets
import sklearn.metrics as sm
import pandas as pd
import numpy as np

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']

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

# Plot the Original Classifications
plt.subplot(1, 3, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')


# Plot the Models Classifications
#KMeans
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(X)
plt.subplot(1, 3, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
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_))

#EM(GMM)
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(X)
y_gmm = gmm.predict(X)
plt.subplot(1, 3, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

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

Output:

```
The accuracy score of K-Mean:  0.24
The Confusion matrixof K-Mean:  [[ 0 50  0]
 [48  0  2]
 [14  0 36]]
The accuracy score of EM:  0.9666666666666667
The Confusion matrix of EM:  [[50  0  0]
 [ 0 45  5]
 [ 0  0 50]]
```



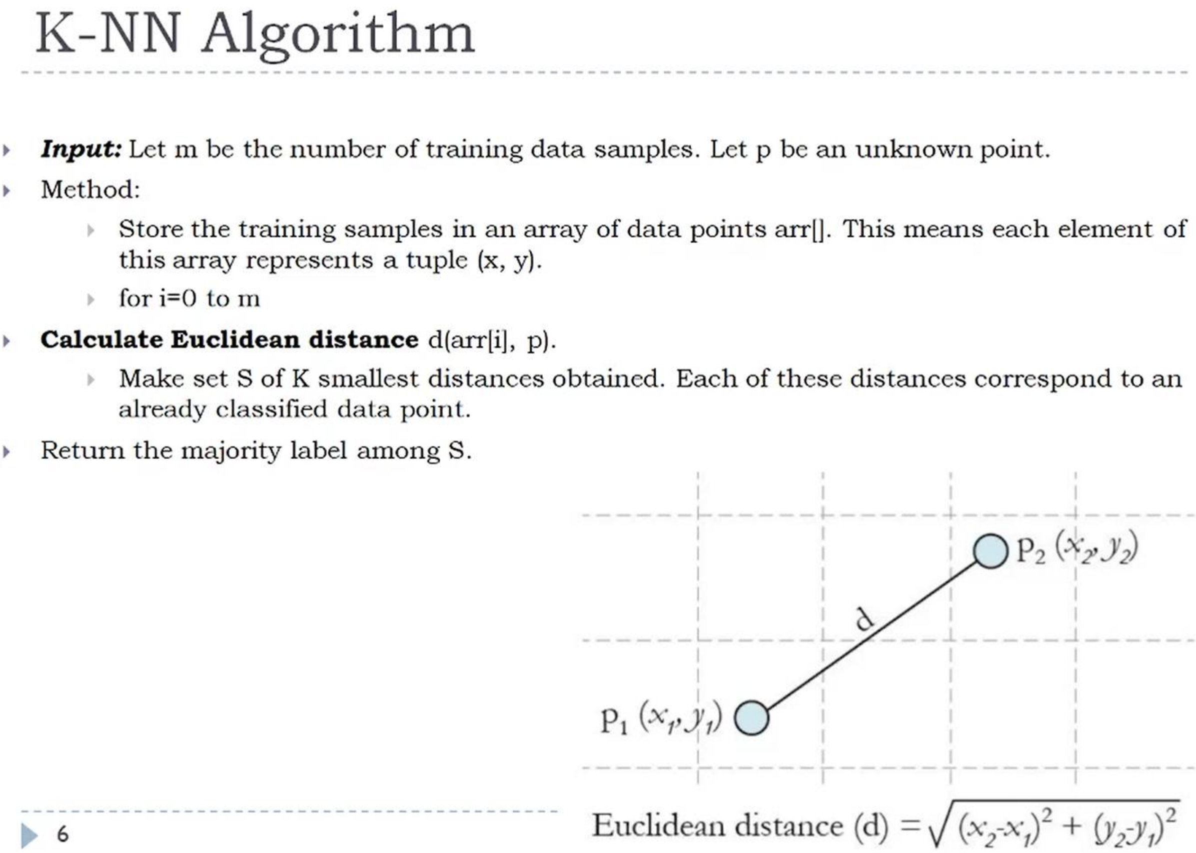
1. EXPERIMENT NO: 9

2. TITLE: **K-NEAREST NEIGHBOUR**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM:
- 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 forthis problem.

5. THEORY:
- K-Nearest Neighbors is one of the most basic yet essential classification algorithms inMachine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.
- It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does notmake any underlying assumptions about the distribution of data.



6. LEARNING OUTCOMES :
- The student will be able to implement k-Nearest Neighbour algorithm to classify the irisdata set and Print both correct and wrong predictions.

7. APPLICATION AREAS:
- Recommender systems
- Classification problems

8. PROGRAMME:

```
from sklearn import datasets
iris=datasets.load_iris()
iris_data=iris.data
iris_labels=iris.target

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(iris_data,iris_labels,test_size=0.30)

from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)

target_names = iris.target_names
for pred,actual in zip(y_pred,y_test):
    print("Prediction is "+str(target_names[pred])+", Actual is "+str(target_names[actual]))

from sklearn.metrics import classification_report,confusion_matrix
print('Confusion matrix is as follows')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

		Predicted			
		Iris-setosa	Iris-versicolor	Iris-virginica	Actual Total
Actual	Iris-setosa	14	0	0	14
	Iris-versicolor	0	13	2	15
	Iris-virginica	0	1	15	16
	Predicted Total	14	14	17	45

Output:

```
Confusion matrix is as follows
[[14  0  0]
 [ 0 13  2]
 [ 0  1 15]]
Accuracy Metrics
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.93      0.87      0.90        15
           2       0.88      0.94      0.91        16

    accuracy                           0.93        45
   macro avg       0.94      0.93      0.94        45
weighted avg       0.93      0.93      0.93        45
```

```
Prediction is virginica, Actual is virginica
Prediction is virginica, Actual is virginica
Prediction is versicolor, Actual is versicolor
Prediction is versicolor, Actual is versicolor
Prediction is setosa, Actual is setosa
Prediction is versicolor, Actual is versicolor
Prediction is versicolor, Actual is versicolor
Prediction is virginica, Actual is virginica
Prediction is virginica, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is versicolor
Prediction is versicolor, Actual is versicolor
Prediction is virginica, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is versicolor, Actual is versicolor
Prediction is virginica, Actual is virginica
Prediction is virginica, Actual is virginica
Prediction is versicolor, Actual is versicolor
Prediction is virginica, Actual is virginica
Prediction is versicolor, Actual is versicolor
Prediction is virginica, Actual is virginica
Prediction is versicolor, Actual is versicolor
Prediction is setosa, Actual is setosa
Prediction is versicolor, Actual is versicolor
Prediction is versicolor, Actual is versicolor
Prediction is versicolor, Actual is versicolor
Prediction is versicolor, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is setosa, Actual is setosa
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is virginica
Prediction is virginica, Actual is virginica
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is virginica
Prediction is virginica, Actual is virginica
Prediction is versicolor, Actual is versicolor
Prediction is setosa, Actual is setosa
Prediction is setosa, Actual is setosa
Prediction is setosa, Actual is setosa
Prediction is setosa, Actual is setosa
Prediction is virginica, Actual is versicolor
```

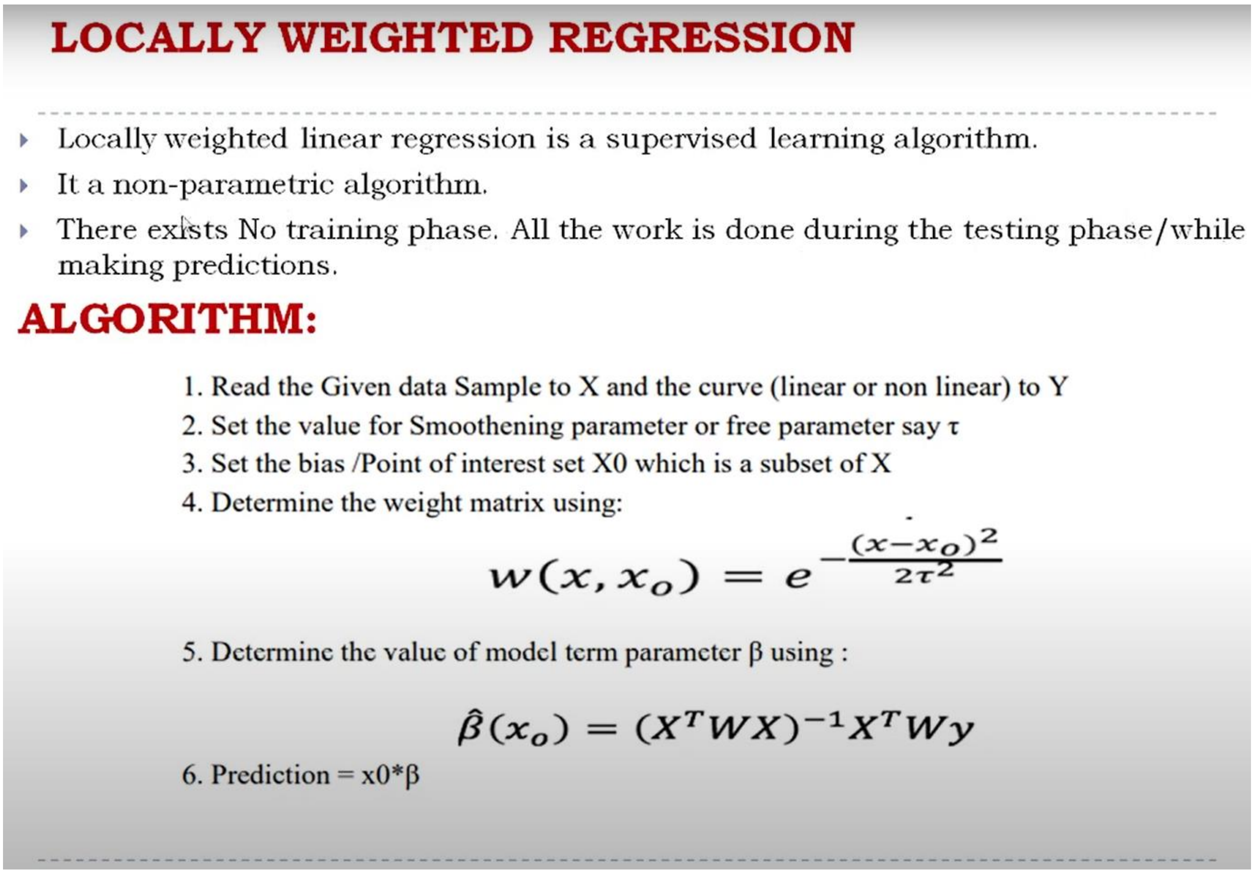
1. EXPERIMENT NO: 10

2. TITLE: **LOCALLY WEIGHTED REGRESSION ALGORITHM**

3. LEARNING OBJECTIVES:
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

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

5. THEORY:



6. LEARNING OUTCOMES :
- To understand and implement linear regression and analyse the results with change in theparameters

7. APPLICATION AREAS:
- Demand anaysis in business
- Optimization of business processes
- Forecasting

8. PROGRAMME:

```python
from numpy import *
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np1
import numpy.linalg as np
from scipy.stats.stats import pearsonr

def kernel(point,xmat, k):
    m,n = np1.shape(xmat)
    weights = np1.mat(np1.eye((m)))
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np1.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 = np1.shape(xmat)
    ypred = np1.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred

# load data points
data = pd.read_csv('D://tips.csv')
bill = np1.array(data.total_bill)
tip = np1.array(data.tip)

#preparing and add 1 in bill
mbill = np1.mat(bill)
mtip = np1.mat(tip) # mat is used to convert to n dimesiona to 2 dimensional array form
m= np1.shape(mbill)[1]
# print(m) 244 data is stored in m
one = np1.mat(np1.ones(m))
X= np1.hstack((one.T,mbill.T)) # create a stack of bill from ONE
#print(X)
#set k here
ypred = localWeightRegression(X,mtip,0.3)
SortIndex = X[:,1].argsort(0)
xsort = X[SortIndex][:,0]

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(bill,tip, color='green')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)
plt.xlabel('Total bill')
plt.ylabel('Tip')
plt.show();
```

Output: