ARUNACHALA COLLEGE OF ENGINEERING FOR WOMEN, MANAVILAI

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

AD3461 MACHINE LEARNING LABORATORY

(Regulation-2021)

Name of the Student: _____

Register Number :_____

Year &Branch :_____

Semester :_____

Subject Code & Name : AD3461-MACHINE LEARNING LABORATORY

Branch : AI&DS

Year/Semester : II / IV

Course Outcomes :

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

Cos	Knowledge Level	Course Outcomes
CO1	**K1**	Apply suitable algorithms for selecting the appropriate features for analysis.
CO2	**K3**	Implement supervised machine learning algorithms on standard datasets and evaluate the performance.
CO3	**K3**	Apply unsupervised machine learning algorithms on standard datasets and evaluate the performance.
CO4	**K3**	Build the graph based learning models for standard data sets.
CO5	**K4**	Assess and compare the performance of different ML algorithms and select the suitable onebased on the application.

COLLEGE VISION & MISSION STATEMENT

Vision

- To inculcate value-based technical education and produce outstanding women graduates to compete with the technological challenges with the right attitude towards social empowerment.

Mission

- To equip resources and establish infrastructure for a beneficial process that paves the way for ideal technocrats.
- To educate and make efficient students with needed skills and make them industry-ready engineers.
- To establish high-level learning and research skills to confront technological scenarios.

To provide valuable resources for social empowerment and lifelong learning process

DEPARTMENT Vision & Mission

Vision

- To produce eminent skilled women engineers in the field of artificial intelligence and Data science to solve challenges in society.

Mission

- To nourish the core competence in the domain of artificial intelligence & data science by continuously improving the resources.
- To produce virtuous engineers by imparting the value of honesty, courage and tenderness to serve the society.
- To enrich the skills to meet the industrial requirement in the field of artificial intelligence.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

PEO1: Develop actionable competency in the realm of Artificial Intelligence & Data Science to solve socially relevant problems.

PEO2: Develop world class solutions with the professional knowledge gained by lifelong learning and work in multi-disciplinary communication. teams with effective.

PEO3: Pursue higher education and research in the field of Artificial Intelligence and Data Science.

Program Outcomes (PO'S)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5.Modern tool usage:

Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write

effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes PROGRAM (PSO'S)

PSO1: Implement Artificial Intelligence and data science tools like data analytics, machine learning and neural networks to build new algorithms for a successful career and entrepreneurship.

PSO2: Apply the knowledge in various fields of healthcare, education, intelligent transportation, and the multidisciplinary field of Artificial Intelligence and Data Science.

PSO3: Graduates will gain practical proficiency with emerging technologies and open source software in the field of Artificial Intelligence and Data Science

LIST OF EXPERIMENTS WITH COs, POs & PSOs

Exp. No.	Name of the Experiment	Cos	POs
1	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.	Co1	1,2,3,4,5,9,10,11,12
2	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.	Co2	1,2,3,4,5,9,10,11,12
3	Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.	Co2	1,2,3,4,5,9,10,11,12
4	Write a program to implement the **naïve Bayesian classifier** for a sample training data set stored as a CSV file and compute the accuracy with a few test data sets.	Co2	1,2,3,4,5,9,10,11,12
5	Implement **naïve Bayesian Classifier** model to classify a set of documents and measure the accuracy, precision, and recall.	Co2	1,2,3,4,5,9,10,11,12
6	Write a program to construct a **Bayesian network to** diagnose CORONA infection using standard WHO Data Set.	Co2	1,2,3,4,5,9,10,11,12
7	Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set forclustering using the k-Means **algorithm**. Compare the results of these two algorithms.	Co3	1,2,3,4,5,9,10,11,12
8	Write a program to implement ***k*-Nearest Neighbour algorithm** to classify the iris data set. Print both correct and wrong predictions.	Co4	1,2,3,4,5,9,10,11,12
9	Implement the non-parametric **Locally Weighted Regression algorithm** in order to fit datapoints. Select an appropriate data set for your experiment and draw graphs.	Co5	1,2,3,4,5,9,10,11,12
ADDITIONAL EXPERIMENTS			
10	Write a program to implement the Decision tree	Co1	1,2,3,4,5,9,10,11,12
11	Write a program to implement 't' test	Co5	1,2,3,4,5,9,10,11,12

INSTRUCTIONS TO STUDENTS

DOs:

- Always sit on assigned computer.
- Enter laboratory in time and work quietly.
- Use the computer properly to keep it in good working condition.
- Wear id_cards and lab coats before entering the laboratory.
- Report the problems identifies in the computer to the staff in charge.
- Shut down the computer properly before leaving the lab.

DON'Ts:

- Do not eat in the lab.
- D not wander around the room and distract other students.
- Do not remove anything from the computer laboratory without permission.
- Avoid stepping on electrical wires or any other computer cables
- Do not change the computer settings.
- Unauthorized access to any software is prohibited.
- Do not connect pen drive or any other storing devices.
- Do not download any software from unauthorized sites.
- Do not send any emails to unauthorized persons without permission.

INDEX

Sl. No.	Date	Topic	Page No	Signature

Ex.No:1.

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.

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.

CANDIDATE-ELIMINATION Learning Algorithm

The CANDIDATE-ELIMINTION algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

Initialize G to the set of maximally general hypotheses in H

Initialize S to the set of maximally specific hypotheses in H

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 any 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

CANDIDATE- ELIMINTION algorithm using version spaces

Training Examples:

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

PROGRAM:

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv("C:\python\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):
        if target[i] == "yes":
         for x in range(len(specific_h)):
```

```
                    if h[x]!= specific_h[x]:
                        specific_h[x] ='?'
                        general_h[x][x] ='?'
                    print(specific_h)
            print(specific_h)
                if target[i] == "no":
                 for x in range(len(specific_h)):
                        if h[x]!= specific_h[x]:
                            general_h[x][x] = specific_h[x]
                        else:
                            general_h[x][x] = '?'
                print(" steps of Candidate Elimination Algorithm",i+1)
                print(specific_h)
                print(general_h)
            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")
```

Data Set:

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

OUTPUT:

[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'high' 'strong' 'warm' 'same']

['rainy' 'cold' 'high' 'strong' 'warm' 'change']

['sunny' 'warm' 'high' 'strong' 'cool' 'change']]

['yes' 'yes' 'no' 'yes']

initialization of specific_h and general_h

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

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

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

steps of Candidate Elimination Algorithm 1

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

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

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

steps of Candidate Elimination Algorithm 2

['sunny' 'warm' '?' 'strong' 'warm' 'same']

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

['sunny' 'warm' '?' 'strong' 'warm' 'same']

steps of Candidate Elimination Algorithm 3

['sunny' 'warm' '?' 'strong' 'warm' 'same']

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' 'warm' 'same']

['sunny' 'warm' '?' 'strong' '?' 'same']

['sunny' 'warm' '?' 'strong' '?' '?']

['sunny' 'warm' '?' 'strong' '?' '?']

steps of Candidate Elimination Algorithm 4

['sunny' 'warm' '?' 'strong' '?' '?']

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific_h:

['sunny' 'warm' '?' 'strong' '?' '?']

Final General_h:

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]

RESULT:

Thus the program to demonstrate the Candidate-Elimination algorithm was implemented and executed successfully.

Ex.No:2.

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.

AIM:

To demonstrate the working of the decision tree based ID3 algorithm with an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

ID3 Algorithm

- 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* $_{vi}$, be the subset of Examples that have value v_i for *A*
    - If *Examples* $_{vi}$, is empty
      - Then below this new branch add a leaf node with label = most common value of Target_attribute in Examples
      - Else below this new branch add the subtree
        ID3(*Examples* $_{vi}$, Targe_tattribute, Attributes – {A}))
- End
- Return Root

ENTROPY:

Entropy measures the impurity of a collection of examples.

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

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

INFORMATION GAIN:

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

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

Training Dataset:

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

Test Dataset:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*
T1	Rain	Cool	Normal	Strong
T2	Sunny	Mild	Normal	Strong

PROGRAM:

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

data = pd.read_csv("C:\python\playtennis.csv")
features = [feat for feat in data]
features.remove("answer")
class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""
def printTree(root: Node, depth=0):
    for i in range(depth):
        print("\t", end="")
    print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    print()
    for child in root.children:
        printTree(child, depth + 1)
def classify(root: Node, new):
    for child in root.children:
        if child.value == new[root.value]:
            if child.isLeaf:
                print ("Predicted Label for new example", new," is:", child.pred)
                exit
            else:
                classify (child.children[0], new)
root = ID3(data, features)
```

```
print("Decision Tree is:")
printTree(root)
print ("------------------")

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

O

OUTPUT:

```
Decision Tree is:
outlook
        overcast ->  ['yes']

        rain
                wind
                        strong ->  ['no']

                        weak ->  ['yes']

        sunny
                humidity
                        high ->  ['no']

                        normal ->  ['yes']

------------------
```

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

RESULT:

Thus the program to demonstrate the working of the decision tree based ID3 algorithm with an appropriate data set for building the decision tree was implemented and executed successfully.

Ex.No:3.

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

AIM:

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

BackpropagationAlgorithm

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each $(\vec{x}, \vec{t})$, in training examples, Do

    Propagate the input forward through the network:

    1. Input the instance $\vec{x}$, 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 output unit k, calculate its error term δ_k

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

3. For each hidden 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

$$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

BACKPROPAGATION (*training_example*, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $(\vec{x}, \vec{t})$, *where* $(\vec{x})$ *is the vector of network input values,* $(\vec{t})$ *and is the vector of target network output values.*
η *is the learning rate (e.g., .05).* n_i, *is the number of network inputs,* n_{hidden} *the number of units in the hidden layer, and* n_{out} *the number of output units.*
The input from unit i into unit j is denoted x_{ji}, *and the weight from unit i to unit j is denoted* w_{ji}

Training Examples:

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

Normalize the input

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

PROGRAM:

```
import numpy as np


X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0) #maximum of X array longitudinally
y = y/100
def sigmoid (x):
    return 1/(1 + np.exp(-x))
```

```
def derivatives_sigmoid(x):
    return x * (1 - x)
epoch=5
lr=0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)
hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_output) *lr
wh += X.T.dot(d_hiddenlayer) *lr
print ("-----------Epoch-", i+1, "Starts----------")
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
print ("-----------Epoch-", i+1, "Ends----------\n")
```

OUTPUT:

Input:

[[0.66666667 1.]

[0.33333333 0.55555556]

[1. 0.66666667]]

Actual Output:

[[0.92]

[0.86]

[0.89]]

Predicted Output:

[[0.86039886]

[0.8461084]

[0.86187001]]

RESULT:

Thus the program to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets was implemented and executed successfully.

Ex.No:4.

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.

AIM:

To write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file and compute the accuracy of the classifier.

Naïve Bayesian Classifier Algorithm

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

PROGRAM

```
import csv
import random
import math


def safe_div(x,y):
    if y == 0:
        return 0
    return x/y


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


def splitdataset(dataset, splitratio):
    trainsize = int(len(dataset) * splitratio);
    trainset = []
```

```
    copy = list(dataset);
    while len(trainset) < trainsize:
        index = random.randrange(len(copy));
        trainset.append(copy.pop(index))
    return [trainset, copy]

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

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

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

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

def summarizebyclass(dataset):
    separated = separatebyclass(dataset);
```

```
    summaries = {}
    for classvalue, instances in separated.items():
        summaries[classvalue] = summarize(instances)
    return summaries

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

def calculateclassprobabilities(summaries, inputvector):
    probabilities = {}
    for classvalue, classsummaries in summaries.items():
        probabilities[classvalue] = 1
        for i in range(len(classsummaries)):
            mean, stdev = classsummaries[i]
            x = inputvector[i]
            calculateprobability(x, mean, stdev);
    return probabilities
def predict(summaries, inputvector):
    probabilities = calculateclassprobabilities(summaries,
inputvector)
    bestLabel, bestProb = None, -1
    for classvalue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classvalue
    return bestLabel

def getpredictions(summaries, testset):
    predictions = []
    for i in range(len(testset)):
```

```
        result = predict(summaries, testset[i])
        predictions.append(result)
    return predictions


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


def main():
    filename='C:\python\prima.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)))
    summaries = summarizeByClass(trainingSet);
    predictions = getPredictions(summaries, testSet)
    accuracy = getaccuracy(testSet, predictions)
    print('Accuracy of the classifier is :{0}%'.format(accuracy))


main()
```

OUTPUT:

```
Split 306 rows into train=205 and test=101 rows
Accuracy: 72.27722772277228%
```

RESULT:

Thus the program to implement the naïve Bayesian classifier for a sample training data set and computing the accuracy of the classifier was executed successfully.

Ex.No:5.

Implement naïve Bayesian Classifier model to classify a set of documents and measure the accuracy, precision, and recall.

AIM:

To implement naïve Bayesian Classifier model to classify a set of documents and measure the accuracy, precision, and recall.

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* ← ***c*** 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 *vj*
     - $P(v_j)$ ← $|docs_j| / |\text{Examples}|$
     - $Text_j$ ← 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 $\boldsymbol{w_k}$ occurs in $Text_j$
       - $P(w_k|v_j)$ ← $(n_k + 1) / (n + |Vocabulary|)$

CLASSIFY_NAIVE_BAYES_TEXT (Doc)

Return the estimated target value for the document Doc. a_i denotes the word found in the i^{th} position within Doc.

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

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

Dataset:

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

PROGRAM:

```
import pandas as pd
msg=pd.read_csv('C:/python/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
print(X)
print(y)

from sklearn.model_selection import train_test_split
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)

from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(count_vect.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

from sklearn import metrics
print('\n Accuracy of the classifer is',
      metrics.accuracy_score(ytest,predicted))

print('\n Confusion matrix')
```

```
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision' ,
metrics.precision_score(ytest,predicted))
print('\n The value of Recall' ,
metrics.recall_score(ytest,predicted))
```

OUTPUT:

```
The dimensions of the dataset (18, 2)
0                       I love this sandwich
1                  This is an amazing place
2         I feel very good about these beers
3                       This is my best work
4                       What an awesome view
5              I do not like this restaurant
6                   I am tired of this stuff
7                     I can't deal with this
8                       He is my sworn enemy
9                        My boss is horrible
10                 This is an awesome place
11    I do not like the taste of this juice
12                             I love to dance
13        I am sick and tired of this place
14                      What a great holiday
15           That is a bad locality to stay
16           We will have good fun tomorrow
17         I went to my enemy's house today
Name: message, dtype: object
0     1
1     1
2     1
3     1
4     1
5     0
6     0
7     0
8     0
9     0
10    1
11    0
12    1
13    0
14    1
15    0
16    1
17    0


The total number of Training Data : (13,)
The total number of Test Data : (5,)
Accuracy of the classifer is 0.8
Confusion matrix
[[3 0]
[1 1]]
 The value of Precision 1.0
 The value of Recall 0.5
```

RESULT:

Thus the program to implement naïve Bayesian Classifier model to classify a set of documents and measure the accuracy, precision, and recall was executed successfully.

Ex.No:6.

Write a program to construct a Bayesian network to diagnose CORONA infection using standard WHO Data set

AIM:

To write a program to construct a Bayesian network to diagnose CORONA infection using standard WHO Data set.

Data set:

Attribute Information:

1. Breathing Problem
2. Fever
3. Dry cough
4. Sore throat
5. Running Nose
6. Asthma
7. Chronic Lung disease
8. Headache
9. Heart disease
10. Diabetes
11. Hyper tension

PROGRAM:

```
import pandas as pd
import numpy as np
import time
import json

def pre_processing(df):
    print("Processing Test Data...")
    time.sleep(2)
```

```
    x = df.drop(df.columns[-1],axis=1)
    y = df[df.columns[-1]]
    print("Done!\n")
    return x, y

def test_model(df,x,y,testData,trained_data):
    print(f"Model Testing In Progress...")
    time.sleep(3)
    total=len(df[y.name])
    totalYes=trained_data["totalYes"]
    totalNo=trained_data["totalNo"]
    sum1Yes=(totalYes/total)
    sum2Yes=1
    for indx2 in range(len(testData)):
        sum1Yes*=(trained_data[x.columns[indx2]][testData[indx2]]['yes']/totalYes)
        sum2Yes*=(trained_data[x.columns[indx2]][testData[indx2]]['total']/total)
    sumYes = sum1Yes/sum2Yes
    sum1No=(totalNo/total)
    sum2No=1
    for indx2 in range(len(testData)):
        sum1No*=(trained_data[x.columns[indx2]][testData[indx2]]['no']/totalNo)
        sum2No*=(trained_data[x.columns[indx2]][testData[indx2]]['total']/total)
    sumNo = sum1No/sum2No
    print(f"\nProbability of Covid 19 to be True : {format(sumYes,'.2f')}")
    print(f'Probability of Covid 19 to be False : {format(sumNo,'.2f')}")
    if sumYes>sumNo:
        ans="Yes"
    else:
        ans="No"
    print(f'Answer is - {ans}\n\n\n")

def read_data():
```

```
    print("Reading data from csv...")
    time.sleep(2)
    df = pd.read_csv("C:/python/covid.csv")
    print("Done...\n")
    return df

def load_model():
    print("Loading Trained Data from trained_data.txt...")
    time.sleep(2)
    with open('C:/python/trained_data.txt') as f:
        data = f.read()
        js = json.loads(data)
    print("Loaded!\n")
    return js

df = pd.DataFrame(read_data())
x,y=pre_processing(df)
x=x.truncate(0,4433)
y=y.truncate(0,4433)
trained_data=load_model()
while True:
    print("\nEnter you choice of features:\n")
    test=[]
    for uniqueCol in list(x.columns):
        columnData=list(np.unique(df[uniqueCol]))
        for idx in range(len(columnData)):
            print(f"{idx}. {columnData[idx]}")
        inp=int(input(f"Enter option for {uniqueCol}: "))
        test.append(columnData[inp])
        print("\n")
    print(f"Your test data is: {test}\n")
    test_model(df,x,y,test,trained_data)
```

OUTPUT:

```
Reading data from csv...
Done...

Processing Test Data...
Done!

Loading Trained Data from trained_data.txt...
Loaded!


Enter you choice of features:

0. No
1. Yes
Enter option for Breathing Problem: 1


0. No
1. Yes
Enter option for Fever: 1


0. No
1. Yes
Enter option for Dry Cough: 1


0. No
1. Yes
Enter option for Sore throat: 1


0. No
1. Yes
Enter option for Running Nose: 0


0. No
1. Yes
Enter option for Asthma: 0


0. No
1. Yes
Enter option for Chronic Lung Disease: 0


0. No
1. Yes
Enter option for Headache: 1
```

```
0. No
1. Yes
Enter option for Heart Disease: 0


0. No
1. Yes
Enter option for Diabetes: 0


0. No
1. Yes
Enter option for Hyper Tension: 0


0. No
1. Yes
Enter option for Abroad travel: 1


0. No
1. Yes
Enter option for Contact with COVID Patient: 0


0. No
1. Yes
Enter option for Attended Large Gathering: 1


0. No
1. Yes
Enter option for Visited Public Exposed Places: 0


0. No
1. Yes
Enter option for Family working in Public Exposed Places: 1


0. No
1. Yes
Enter option for Wearing Masks: 1


0. No
1. Yes
Enter option for Sanitization from Market: 1


Your test data is: ['Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes'
, 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes']

Model Testing In Progress...
Probability of Covid 19 to be True : 0.92
Probability of Covid 19 to be False : 0.00
Answer is - Yes
```

RESULT:

Thus the program to construct a Bayesian network to diagnose CORONA infection was implemented and executed successfully.

Ex.No:7

Apply EM ALGORITHM to classify a set of data stored in a .CSV file. Use the same data set for clustering using the k-means algorithm. Compare the results of these two algorithms.

AIM:

To write a program for EM ALGORITHM to classify a set of data stored in a .CSV file and use the same data set for clustering using the k-means algorithm.

EM ALGORITHM

These are the two basic steps of the EM algorithm, namely **E Step or Expectation Step or Estimation Step** and **M Step or Maximization Step**.

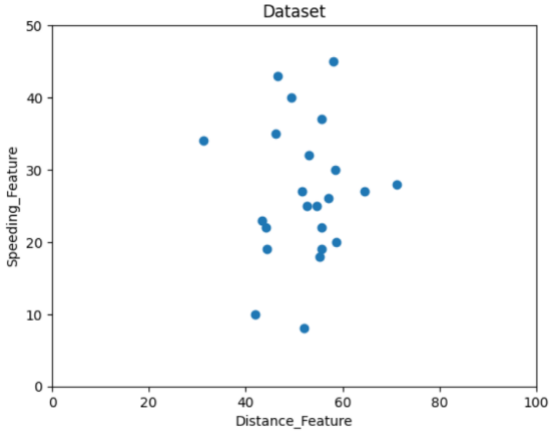
- **Estimation step:**
  - initialize μ_k,$\sum k$ and $\pi_k$by some random values, or by K means clustering results or by hierarchical clustering results.
  - Then for those given parameter values, estimate the value of the latent variables (i.e γ_k)
- **Maximization Step:**
  - Update the value of the parameters(i.e. μ_k,$\sum k$ and π_k) calculated using ML method.

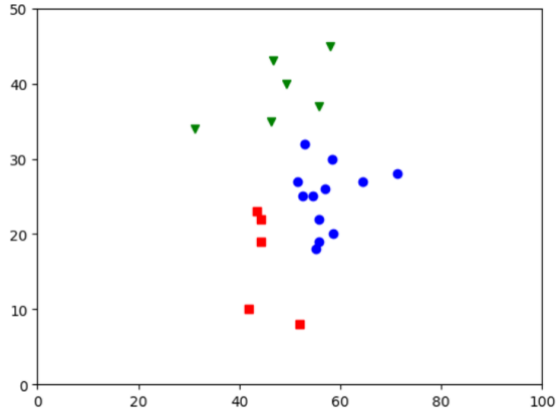
1. Load data set
2. Initialize the mean μ_k, the covariance matrix $\sum k$ and the mixing coefficients
    1. $\pi_k$by some random values. (or other values)
3. Compute the γ_k values for all k.
4. Again Estimate all the parameters using the current γ_k values.
5. Compute log-likelihood function.
6. Put some convergence criterion
7. If the log-likelihood value converges to some value (or if all the parameters converge to some values) then **stop**, else return to **Step 3**.

PROGRAM

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
data = pd.read_csv('C:\python\kmeansdata.csv')
df = pd.DataFrame(data)
print(df)
col1 = df['Distance_Feature']
col2 = df['Speeding_Feature']
print(list(zip(col1,col2)))
X = np.matrix(list(zip(col1,col2)))
plt.xlim([0, 100])
plt.ylim([0, 50])
plt.title('Dataset')
plt.ylabel('Speeding_Feature')
plt.xlabel('Distance_Feature')
plt.scatter(col1,col2)
plt.plot()
plt.show()
colors = ['b','g','r']
markers = ['o','v','s']
kmeans_model = KMeans(n_clusters=3).fit(X)
plt.plot()
for i,l in enumerate(kmeans_model.labels_):
    plt.plot(col1[i], col2[i], color=colors[l], marker=markers[l], ls='None')
    plt.xlim([0, 100])
    plt.ylim([0, 50])
plt.show()
```

OUTPUT


Dataset
Speeding_Feature
Distance_Feature
0
10
20
30
40
50
0
20
40
60
80
100


0
10
20
30
40
50
0
20
40
60
80
100

RESULT:

Thus the program for EM ALGORITHM to classify a set of data stored in a .CSV file was implemented and executed successfully**.**

Ex.No:7.

Write a program to implement k- Nearest Neighbour algorithm to classify the iris data set.

AIM:

To write a program to implement k- Nearest Neighbour algorithm to classify the iris data set.

K-Nearest Neighbour Algorithm

Training algorithm:

- For each training example (x, f (x)), add the example to the list training examples

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from training examples that are nearest to x_q
  - Return

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

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

Data Set:

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

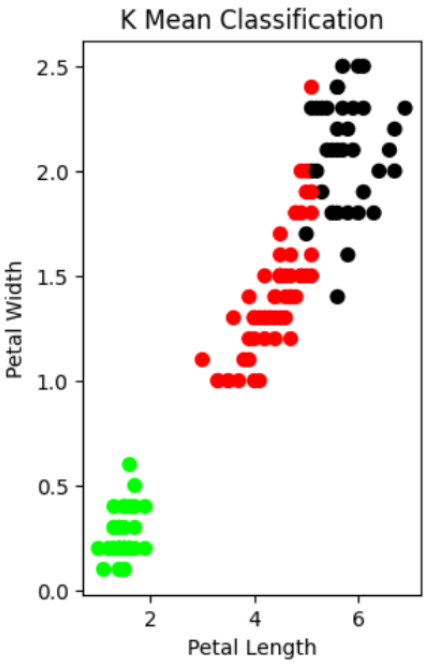
	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

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
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']
model = KMeans(n_clusters=3)
model.fit(X)
plt.figure(figsize=(14,7))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(1, 2, 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')
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
```

OUTPUT:

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



RESULT:

Thus the program to implement k- Nearest Neighbour algorithm to classify the iris data set was executed successfully.

Ex.No:9

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

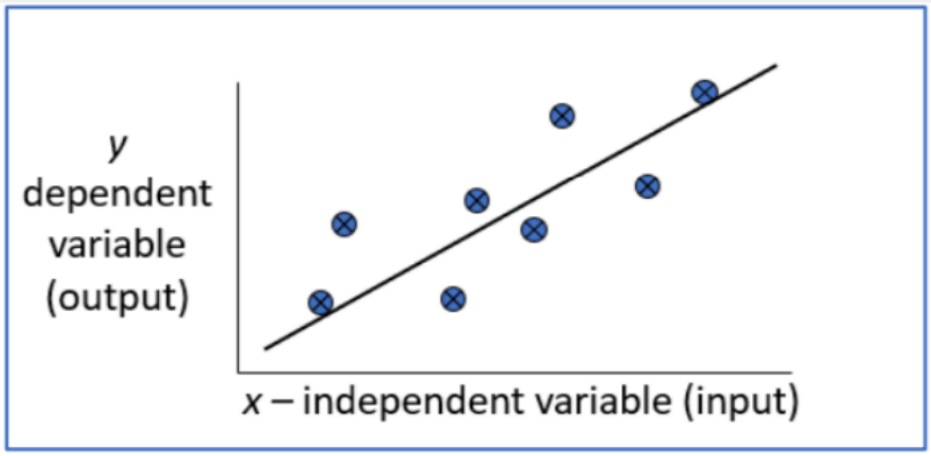
AIM:

To implement the non-parametric Locally Weighted Regression algorithm in order to fit data points and select appropriate data set for your experiment and draw graphs.

Locally Weighted Regression Algorithm

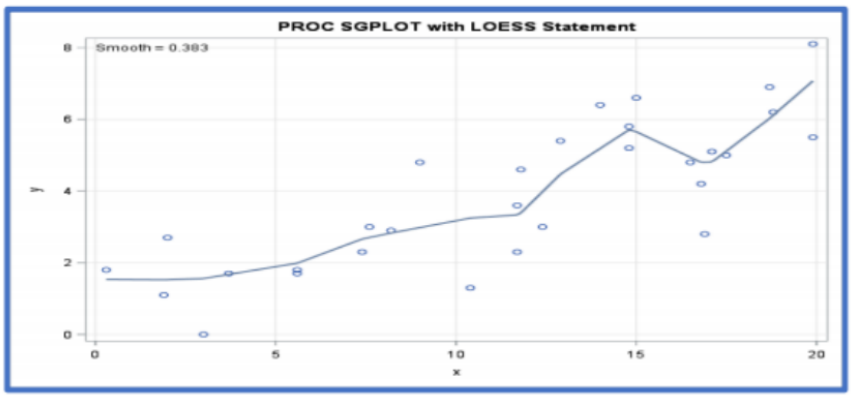
Regression:

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



Loess/Lowess Regression:

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



Lowess Algorithm:

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

ALGORITHM

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

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

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

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

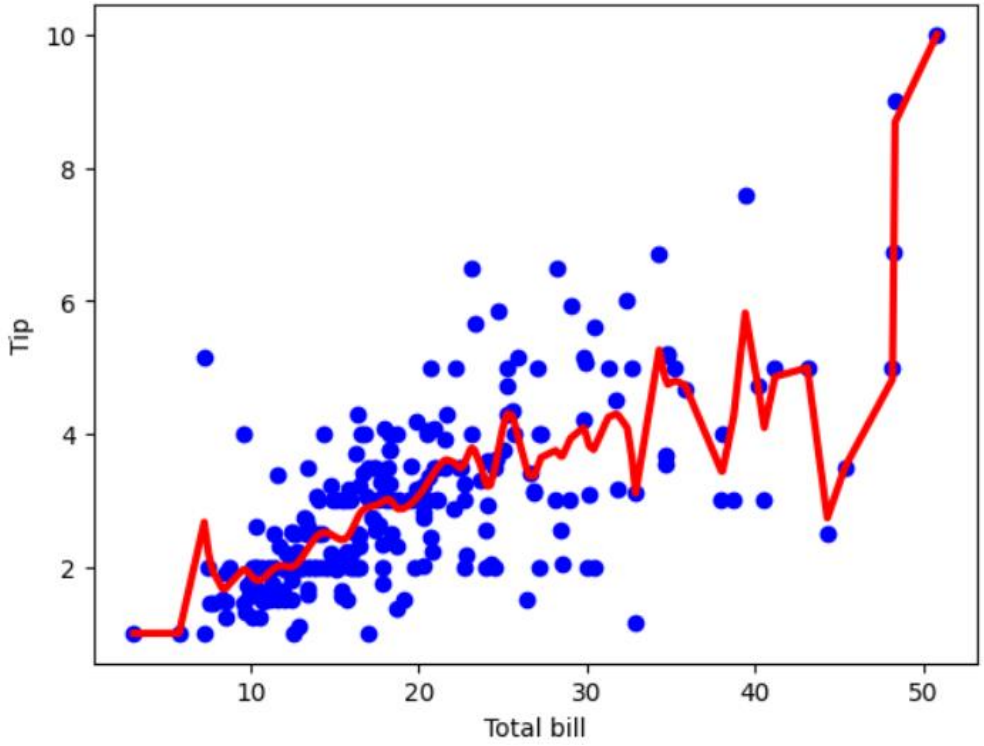
6. Prediction = x0*β:

PROGRAM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_csv('C:/python/tips.csv')
features = np.array(df.total_bill)
labels = np.array(df.tip)


def kernel(data, point, xmat, k):
  m,n = np.shape(xmat)
  ws = np.mat(np.eye((m)))
  for j in range(m):
    diff = point - data[j]
    ws[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
  return ws
```

```
def local_weight(data, point, xmat, ymat, k):
   wei = kernel(data, point, xmat, k)
   return (data.T*(wei*data)).I*(data.T*(wei*ymat.T))

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

   m = features.shape[0]
   mtip = np.mat(labels)
   data = np.hstack((np.ones((m, 1)), np.mat(features).T))
   ypred = local_weight_regression(data, mtip, 0.5)
   indices = data[:,1].argsort(0)
   xsort = data[indices][:,0]
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.scatter(features, labels, color='blue')
    ax.plot(xsort[:,1],ypred[indices], color = 'red', linewidth=3)
    plt.xlabel('Total bill')
    plt.ylabel('Tip')
    plt.show()
```

OUTPUT:



RESULT:

Thus the program to implement the non-parametric Locally Weighted Regression algorithm in order to fit data points and select appropriate data set for your experiment and draw graphs was executed successfully.