KGiSL Institute of Technology

(Approved by AICTE, New Delhi; Affiliated to Anna University, Chennai)

Recognized by UGC, Accredited by NBA (IT)

365, KGiSL Campus, Thudiyalur Road, Saravanampatti, Coimbatore – 641035.

AL3461 - MACHINE LEARNING

NAME : ....

REG. NO. : ....

COURSE : ....

SEMESTER : ....

BATCH : ....

KGiSL Institute of Technology

(Approved by AICTE, New Delhi; Affiliated to Anna University, Chennai)

Recognized by UGC, Accredited by NBA (IT)

365, KGiSL Campus, Thudiyalur Road, Saravanampatti, Coimbatore – 641035.

NAME :

CLASS :

UNIVERSITY REG NO :

Certified that, this is a bonafide record of work done by ………………………………………..

Of ……………………………………………………………….. branch in **MACHINE LEARNING LABORATORY**, during fourth semester of academic year 2022-2023.

Faculty In-charge **Head of the Department**

Submitted during Anna University Practical Examination held on …………………… at KGiSL Institute of Technology, Coimbatore – 641 035.

Internal examiner **External Examiner**

S.NO	DATE	LIST OF THE EXPERIMENTS	PAGE NO	MARKS	SIGNATURE
1		Canidate–Elimination Algorithm			
2		ID3 Algorithm			
3		Backpropagation Algorithm			
4		Naïve Bayesian Classifier			
5		Naïve Bayesian Classifier			
6		Bayesian Network Classifier			
7		Expectation Maximization Algorithm			
8		K-Nearest Neighbour Algorithm			
9		Locally Weighted Regression Algorithm			

EX NO : DATE :	**CANDIDATE ELIMINATION ALGORITHM**

AIM :

To implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

ALGORITHM :

Step1: Load Data set

Step2: Initialize General Hypothesis and Specific Hypothesis.

Step3: For each training example

Step4: If example is positive example

if attribute_value == hypothesis_value:

Do nothing

else:

replace attribute value with '?' (Basically generalizing it)

Step5: If example is Negative example

Make generalize hypothesis more specific.

PROGRAM :

```
import numpy as np
import pandas as pd

data = pd.read_csv('3-dataset.csv')
concepts = np.array(data.iloc[:,0:-1])
print("\nInstances are:\n",concepts)
target = np.array(data.iloc[:,-1])
print("\nTarget Values are: ",target)

def learn(concepts, target):
```

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

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

        if target[i] == "no":
            print("Instance is Negative ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'

        print("Specific Bundary after ", i+1, "Instance is ", specific_h)
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
```

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

WEATHER DATASET :

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

OUTPUT :

```
Instances are:
 [['sunny' 'hot' 'high' 'weak']
 ['sunny' 'hot' 'high' 'strong']
 ['overcast' 'hot' 'high' 'weak']
 ['rain' 'mild' 'high' 'weak']
 ['rain' 'cool' 'normal' 'weak']
 ['rain' 'cool' 'normal' 'strong']
 ['overcast' 'cool' 'normal' 'strong']
 ['sunny' 'mild' 'high' 'weak']
```

['sunny' 'cool' 'normal' 'weak']

['rain' 'mild' 'normal' 'weak']

['sunny' 'mild' 'normal' 'strong']

['overcast' 'mild' 'high' 'strong']

['overcast' 'hot' 'normal' 'weak']

['rain' 'mild' 'high' 'strong']]

Target Values are: ['no' 'no' 'yes' 'yes' 'yes' 'no' 'yes' 'no' 'yes' 'yes' 'yes' 'yes' 'yes' 'no']

Initialization of specific_h and genearal_h

Specific Boundary: ['sunny' 'hot' 'high' 'weak']

Generic Boundary: [['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?']]

Instance 1 is ['sunny' 'hot' 'high' 'weak']

Instance is Negative

Specific Boundary after 1 Instance is ['sunny' 'hot' 'high' 'weak']

Generic Boundary after 1 Instance is [['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?','?']]

Instance 2 is ['sunny' 'hot' 'high' 'strong']

Instance is Negative

Specific Boundary after 2 Instance is ['sunny' 'hot' 'high' 'weak']

Generic Boundary after 2 Instance is [['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', 'weak']]

Instance 3 is ['overcast' 'hot' 'high' 'weak']

Instance is Positive

Specific Boundary after 3 Instance is ['?' 'hot' 'high' 'weak']

Generic Boundary after 3 Instance is [['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', 'weak']]

Instance 4 is ['rain' 'mild' 'high' 'weak']

Instance is Positive

Specific Boundary after 4 Instance is ['?' '?' 'high' 'weak']

Generic Boundary after 4 Instance is [['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', '?'], ['?', '?', '?', 'weak']]

Final Specific_h:

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

Final General_h:

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

RESULT :

Thus the above program to implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples has been executed successfully and the output is verified.

EX NO : DATE :	**ID3 ALGORITHM**

AIM :

To 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.

ALGORITHM :

Step1 : Loading *csv* data in python, (using *pandas* library)

Step2 : Training and building **Decision tree** using **ID3 algorithm**

Step3 : Predicting from the tree

Step4 : Finding out the accuracy

PROGRAM :

```
import pandas as pd
import math
import numpy as np
from google.colab import files
files.upload()
data = pd.read_csv("3-dataset.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 entropy(examples):
    pos = 0.0
    neg = 0.0
    for _, row in examples.iterrows():
        if row["answer"] == "yes":
            pos += 1
        else:
            neg += 1
    if pos == 0.0 or neg == 0.0:
        return 0.0
    else:
        p = pos / (pos + neg)
        n = neg / (pos + neg)
        return -(p * math.log(p, 2) + n * math.log(n, 2))
def info_gain(examples, attr):
    uniq = np.unique(examples[attr])
    #print ("\n",uniq)
    gain = entropy(examples)
    #print ("\n",gain)
    for u in uniq:
```

```
        subdata = examples[examples[attr] == u]
        #print ("\n",subdata)
        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
        #print ("\n",gain)
    return gain
def ID3(examples, attrs):
    root = Node()
    max_gain = 0
    max_feat = ""
    for feature in attrs:
        #print ("\n",examples)
        gain = info_gain(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat
    #print ("\nMax feature attr",max_feat)
    uniq = np.unique(examples[max_feat])
    #print ("\n",uniq)
    for u in uniq:
        #print ("\n",u)
        subdata = examples[examples[max_feat] == u]
        #print ("\n",subdata)
        if entropy(subdata) == 0.0:
```

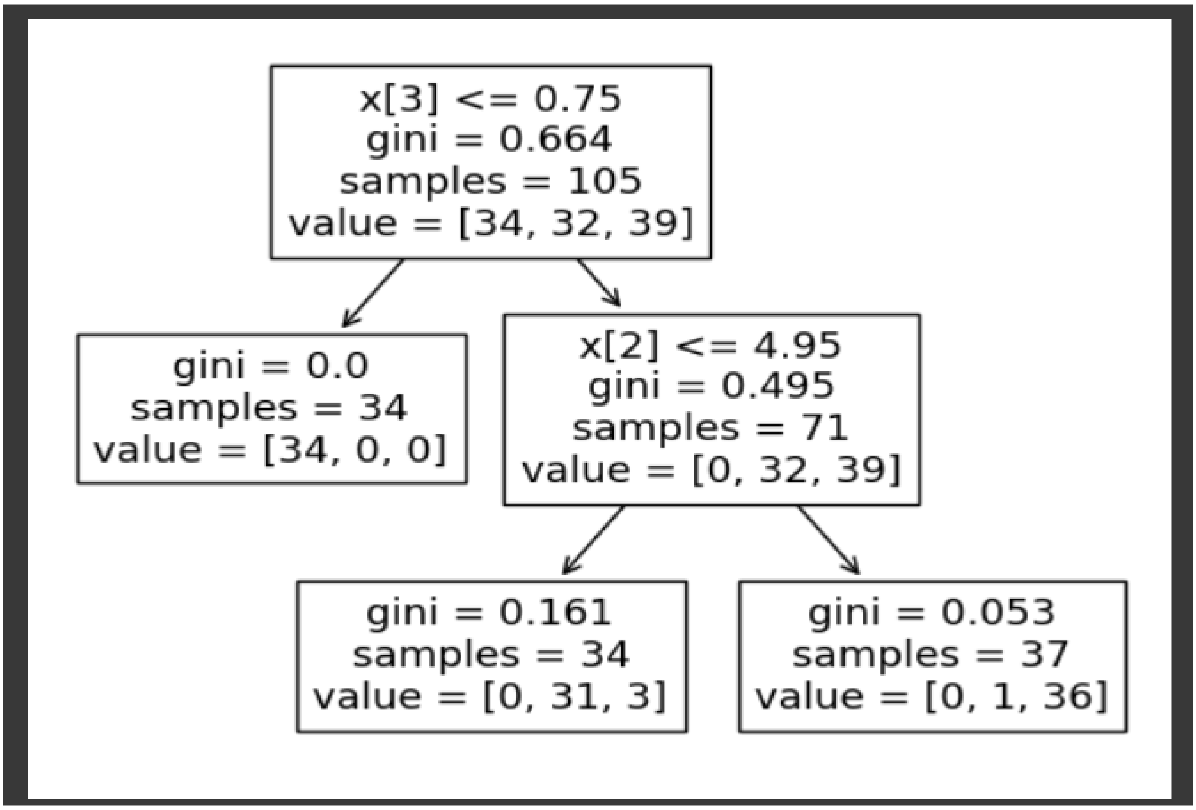
```
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["answer"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)
    return root
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)
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
from sklearn.tree import export_text
clf = DecisionTreeClassifier(random_state=0,max_depth=2)
iris = load_iris()
iris
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=0)
clf.fit(X_train,y_train)
plot_tree(clf)
```

```
r = export_text(clf, feature_names=iris['feature_names'])

print(r)
```

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']

|--- petal width (cm) <= 0.75

|   |--- class: 0

|--- petal width (cm) >  0.75

|   |--- petal length (cm) <= 4.95

|   |   |--- class: 1

|   |--- petal length (cm) >  4.95

|   |   |--- class: 2
```



RESULT :

Thus the above program to demonstrate the working of the decision tree based on ID3 algorithm has been executed successfully and the output is verified.

EX NO : DATE :	**BACKPROPAGATION ALGORITHM**

AIM :

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

ALGORITHM :

Step1 : The input layer receives the input.

Step2 : The input is then averaged overweights.

Step3 : Each hidden layer processes the output.

Step4 : In this step, the algorithm moves back to the hidden layers again to optimize the weights and reduce the error.

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)
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))

for i in range(epoch):
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp= outinp1+bout
    output = sigmoid(outinp)

    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")
```

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

import matplotlib.pyplot as plt
import numpy as np
x = np.array([2,1,3])
y = np.array([9,5,6])
plt.scatter(x,y)
plt.show()
```

OUTPUT :

```
————Epoch- 1 Starts———-
Input:
[[0.66666667 1. ]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]
[0.89]]
Predicted Output:
[[0.81951208]
[0.8007242 ]
[0.82485744]]
————Epoch- 1 Ends———-
```

————Epoch- 2 Starts————-

Input:

[[0.66666667 1.]

[0.33333333 0.55555556]

[1. 0.66666667]]

Actual Output:

[[0.92]

[0.86]

[0.89]]

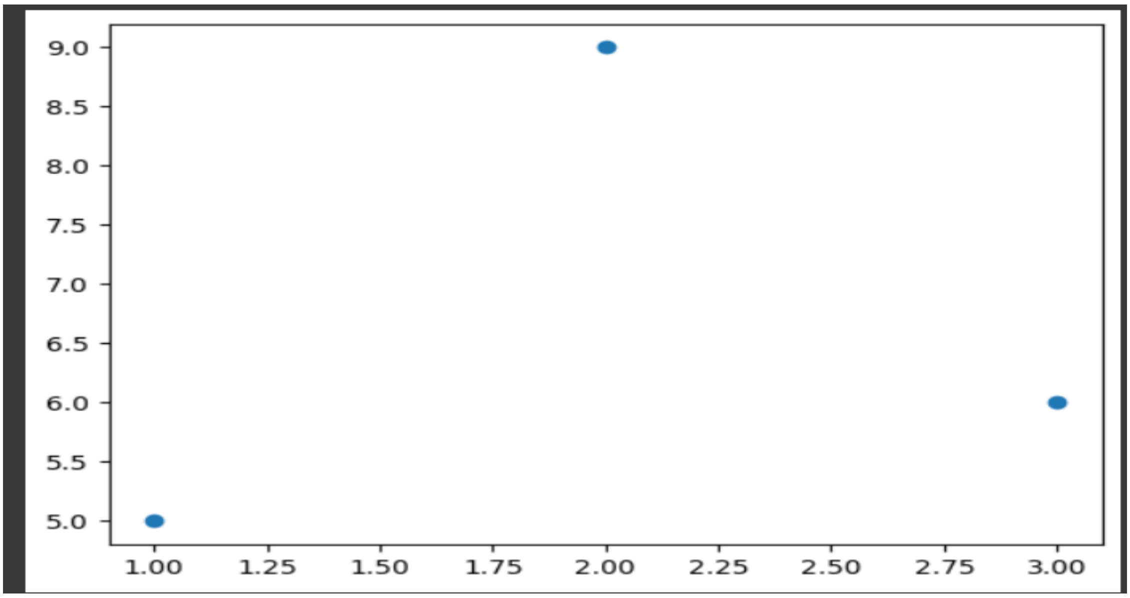
Predicted Output:

[[0.82033938]

[0.80153634]

[0.82568134]]

————Epoch- 2 Ends————-



RESULT :

Thus the above program to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets has been executed successfully and the output is verified.

EX NO : DATE :	**NAÏVE BAYESIAN CLASSIFIER**

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 with a few test data sets.

ALGORITHM :

Step1 : Calculate the prior probability for given class labels.

Step2 : Find Likelihood probability with each attribute for each class.

Step3 : Put these value in Bayes Formula and calculate posterior probability.

Step4 : See which class has a higher probability, given the input belongs to the higher probability class.

WEATHER DATASET :

OUTLOOK	TEMPERATURE	HUMIDITY	WIND	ANSWER
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

PROGRAM :

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
from sklearn import metrics
print("Accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

OUTPUT :

Accuracy : 95.0

RESULT :

Thus the above 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 has been executed successfully and the output is verified.

EX NO : DATE :	**NAÏVE BAYESIAN CLASSIFIER**

AIM :

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

ALGORITHM :

Step1 : Calculate the prior probability for given class labels.

Step2 : Find Likelihood probability with each attribute for each class.

Step3 : Put these value in Bayes Formula and calculate posterior probability.

Step4 : See which class has a higher probability, given the input belongs to the higher probability class.

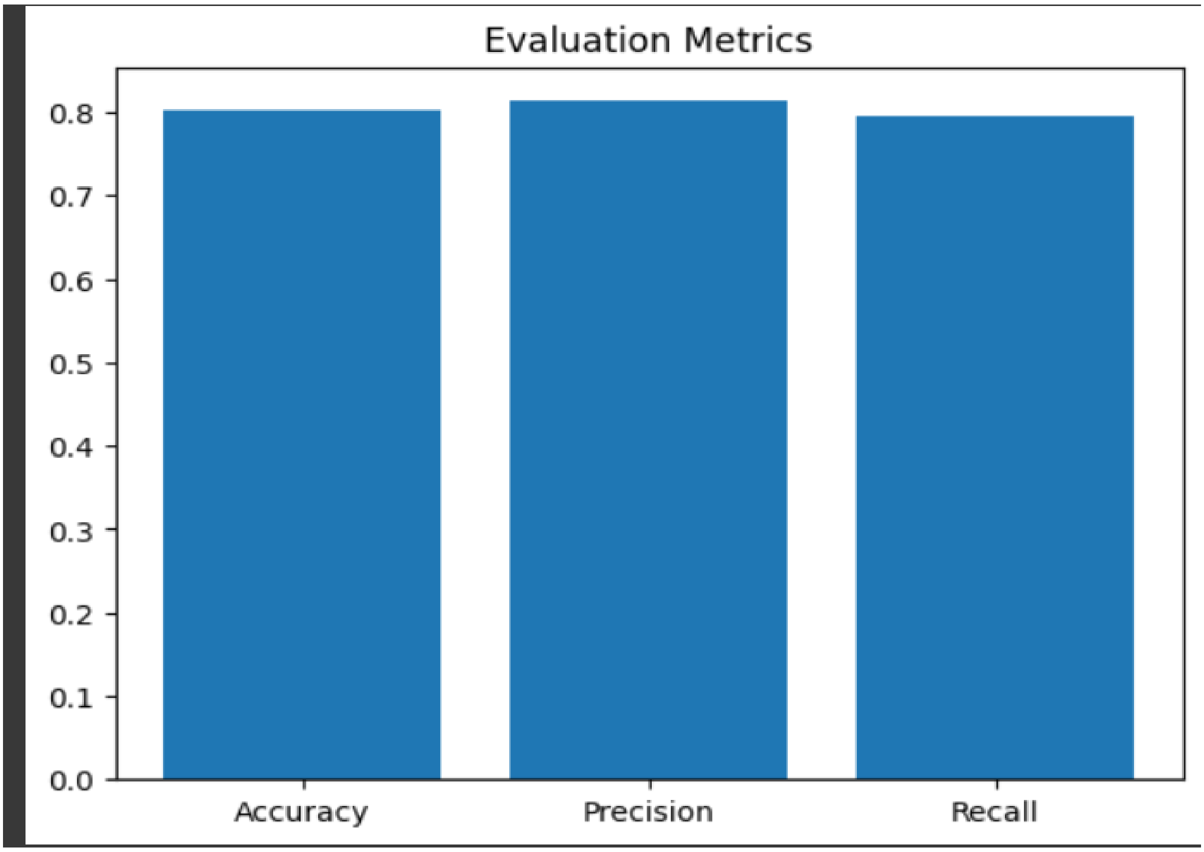
PROGRAM :

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, precision_score, recall_score

newsgroups_train = fetch_20newsgroups(subset='train')

newsgroups_test = fetch_20newsgroups(subset='test')

vectorizer = CountVectorizer(stop_words='english')

X_train = vectorizer.fit_transform(newsgroups_train.data)

X_test = vectorizer.transform(newsgroups_test.data)

nb = MultinomialNB()

nb.fit(X_train, newsgroups_train.target)

y_pred = nb.predict(X_test)
```

```
accuracy = accuracy_score(newsgroups_test.target, y_pred)
precision = precision_score(newsgroups_test.target, y_pred, average='macro')
recall = recall_score(newsgroups_test.target, y_pred, average='macro')
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

import matplotlib.pyplot as plt
accuracy = accuracy_score(newsgroups_test.target, y_pred)
precision = precision_score(newsgroups_test.target, y_pred, average='macro')
recall = recall_score(newsgroups_test.target, y_pred, average='macro')
labels = ['Accuracy', 'Precision', 'Recall']
scores = [accuracy, precision, recall]
plt.bar(labels, scores)
plt.title('Evaluation Metrics')
plt.show()
```

OUTPUT :

```
Accuracy: 0.8023
Precision: 0.8130
Recall: 0.7942
```



RESULT :

Thus the above program to implement naïve Bayesian Classifier model to classify a set of documents and measure the accuracy, precision, and recall has been executed successfully and the output is verified.

EX NO : DATE :	**BAYESIAN NETWORK**

AIM :

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

ALGORITHM :

Step1 : First download the datasets in the .csv file format.

Step2 : Start the program and identify the target variable.

Step3 : Specify the conditional probability tables.

Step4 : Predict the output with high accuracy.

COVID DATASET :

S.NO	OBSERVATION	PROVINCE/STATE	COUNTRY/REGION	LAST UPDATE	CONFIRMED
1	01/22/2020	Anhui	Mainland China	1/22/2020	1
2	01/22/2020	Beijing	Mainland China	1/22/2020	14
3	01/22/2020	Chongqing	Mainland China	1/22/2020	6
4	01/22/2020	Fujian	Mainland China	1/22/2020	1
5	01/22/2020	Gansu	Mainland China	1/22/2020	0
6	01/22/2020	Guangdong	Mainland China	1/22/2020	26
7	01/22/2020	Guangxi	Mainland China	1/22/2020	2

PROGRAM :

```
import pandas as pd
import numpy as np
from sklearn.mixture import GaussianMixture
data = pd.read_csv('covid_19_data.csv')
data = data.select_dtypes(include=[np.number])
em = GaussianMixture(n_components=3)
em.fit(data)
labels = em.predict(data)
print(labels)
```

OUTPUT :

```
[ 0 0 0 . . . 0 1 2 ]
```

RESULT :

Thus the above program to construct a Bayesian network to diagnose CORONA infection using standard WHO Data Set has been executed successfully and the output is verified.

EX NO : DATE :	**EXPECTATION MAXIMIZATION ALGORITHM**

AIM :

To write a program to apply EM algorithm to cluster a set of data stored in a .CSV file and use the same data set for clustering using the k-Means algorithm and then compare the results of these two algorithms.

ALGORITHM :

Step1 : First download the datasets in the .csv file format.

Step2 : Then initialize the parameter values.

Step3 : Then estimate or guess the values of missing data.

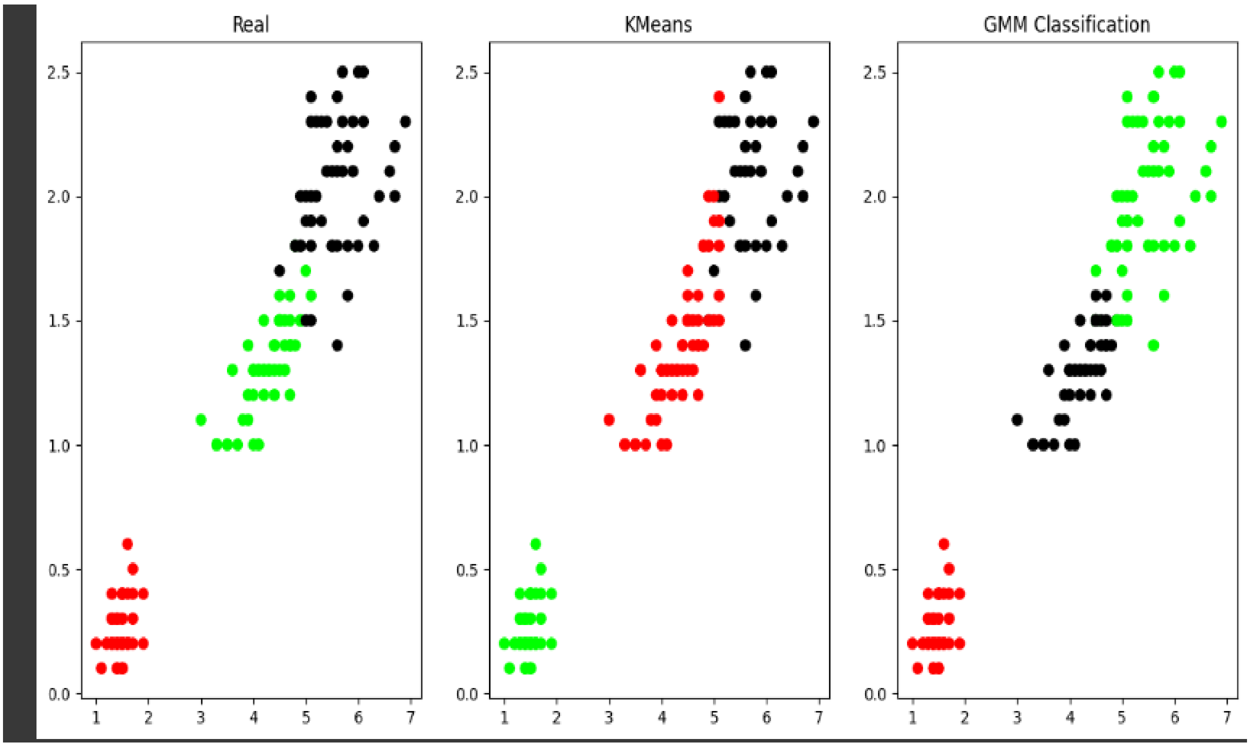
Step4 : Now check the values of latent variables whether it is converging or not and then stop the process.

PROGRAM :

```
from sklearn.cluster import KMeans

from sklearn import preprocessing

from sklearn.mixture import GaussianMixture

from sklearn.datasets import load_iris

import sklearn.metrics as sm

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset=load_iris()

X=pd.DataFrame(dataset.data)
```

```
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y=pd.DataFrame(dataset.target)

y.columns=['Targets']

plt.figure(figsize=(14,7))

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

plt.subplot(1,3,1)

plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)

plt.title('Real')

plt.subplot(1,3,2)

model=KMeans(n_clusters=3)

model.fit(X)

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

plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)

plt.title('KMeans')

scaler=preprocessing.StandardScaler()

scaler.fit(X)

xsa=scaler.transform(X)

xs=pd.DataFrame(xsa,columns=X.columns)

gmm=GaussianMixture(n_components=3)

gmm.fit(xs)

y_cluster_gmm=gmm.predict(xs)

plt.subplot(1,3,3)

plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)

plt.title('GMM Classification')
```

OUTPUT :



RESULT :

Thus the above program to apply EM algorithm has been executed successfully and the output is verified.

EX NO : DATE :	**K-NEAREST NEIGHBOUR ALGORITHM**

AIM :

To write a program to implement k-Nearest Neighbour algorithm to classify the iris data set and print both correct and wrong predictions.

ALGORITHM :

Step1 : Create feature and target variables.
Step2 : Split data into training and test data.
Step3 : Generate a k-NN model using neighbour value.
Step4 : Train or fit the data into the model.
Step5 : Predict the output.

IRIS DATASET :

5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa

PROGRAM :

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

```
from sklearn import metrics
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv("8-dataset.csv", names=names)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
print(X.head())
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.10)
classifier = KNeighborsClassifier(n_neighbors=5).fit(Xtrain, ytrain)
ypred = classifier.predict(Xtest)
i = 0
print ("\n----------------------------------------------------------------------")
print ('%-25s %-25s %-25s' % ('Original Label', 'Predicted Label', 'Correct/Wrong'))
print ("----------------------------------------------------------------------")
for label in ytest:
    print ('%-25s %-25s' % (label, ypred[i]), end="")
    if (label == ypred[i]):
        print (' %-25s' % ('Correct'))
    else:
        print (' %-25s' % ('Wrong'))
    i = i + 1
print ("----------------------------------------------------------------------")
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print ("----------------------------------------------------------------------")
print("\nClassification Report:\n",metrics.classification_report(ytest, ypred))
print ("----------------------------------------------------------------------")
print('Accuracy of the classifer is %0.2f' % metrics.accuracy_score(ytest,ypred))
print ("----------------------------------------------------------------------")
```

OUTPUT :

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

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

```
Confusion Matrix:
 [[7 0 0]
 [0 3 0]
 [0 0 5]]
-------------------------------------------------------------------------

Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      1.00      1.00         3
 Iris-virginica       1.00      1.00      1.00         5

       accuracy                           1.00        15
      macro avg       1.00      1.00      1.00        15
   weighted avg       1.00      1.00      1.00        15

-------------------------------------------------------------------------
Accuracy of the classifer is 1.00
-------------------------------------------------------------------------
```

RESULT :

Thus the above program to implement k-Nearest Neighbour algorithm to classify the iris data set and print both correct and wrong predictions has been executed successfully and the output is verified.

EX NO : DATE :	**LOCALLY WEIGHTED REGRESSION ALGORITHM**

AIM :

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

ALGORITHM :

Step1 : Read the Given data Sample to X and the curve (linear or non linear) to Y.

Step2 : Set the value for Smoothening parameter or Free parameter say τ.

Step3 : Set the bias /Point of interest set x0 which is a subset of X.

Step4 : Determine the weight matrix.

Step5 : Determine the value of model term parameter β.

RESTAURANT BILL DATASET :

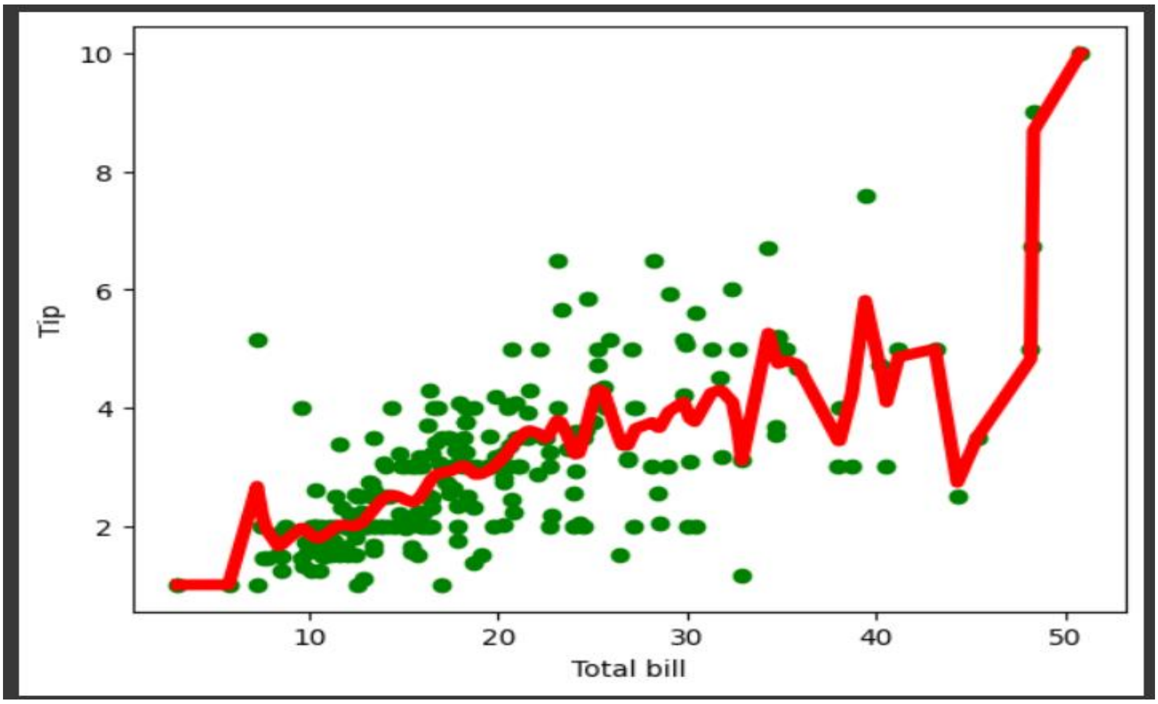
TOTAL_BILL	TIP	SEX	SMOKER	DAY	TIME	SIZE
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2

PROGRAM :

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
def kernel(point, xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m)))
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights
def localWeight(point, xmat, ymat, k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W
def localWeightRegression(xmat, ymat, k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred
data = pd.read_csv('10-dataset.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)
mbill = np.mat(bill)
mtip = np.mat(tip)
```

```
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T))
ypred = localWeightRegression(X,mtip,0.5)
SortIndex = X[:,1].argsort(0)
xsort = X[SortIndex][:,0]
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(bill,tip, color='green')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)
plt.xlabel('Total bill')
plt.ylabel('Tip')
plt.show();
```

OUTPUT :



RESULT :

Thus the above program to implement the non-parametric locally weighted regression algorithm has been executed successfully and the output is verified.