DON BOSCO INSTITUTE OF TECHNOLOGY
Kumbalagodu, Mysore Road,Bengaluru–560074

Department Of CSE (Artificial Intelligence and machine Learning)

MACHINE LEARNING LABORATORY
(21AIL66)

Prepared by: Dr. B. Kursheed

Academic Year 2023-2024

VI SEMESTER

Machine learning

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

Machine learning tasks

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

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

Semi-supervised learning: the computer is given only an incomplete training

signal: a training set with some (often many) of the target outputs missing.

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

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

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

Supervised Learning	Unsupervised Learning	Semi-supervised Learning
Find- S algorithm	EM algorithm	Locally weighted Regression algorithm
Candidate elimination algorithm	K means algorithm	
Decision tree algorithm		
Back propagation Algorithm		
Naïve Bayes Algorithm		
K nearest neighbour algorithm(lazy learning algorithm)		

Machine learning applications

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

In **regression,** also a supervised problem, the outputs are continuous rather than discrete.

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

Density estimation finds the distribution of inputs in some space.

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

MACHINE LEARNING APPROACHES

- **Decision tree learning**
- **Association rule learning**
- **Artificial neural networks**
- **Deep learning**
- **Inductive logic programming**
- **Support vector machines**
- **Clustering**
- **Bayesian networks**
- **Reinforcement learning**
- **Similarity and metric learning**
- **Genetic algorithms**
- **Rule-based machine learning**
- **Feature selection approach**

MACHINE LEARNING LABORATORY

[As per Choice Based Credit System (CBCS) scheme]

(Effective from the academic year 2024 -2025) SEMESTER – VI

Subject Code	21AIL66	IA Marks	50
Number of Lecture Hours/Week	02P	Exam Marks	100
Total Number of Lecture Hours	42	Exam Hours	03

CREDITS – 01

Course Learning Objectives:

CLO 1. To learn and understand the Importance Machine learning Algorithms

CLO 2. Compare and contrast the learning techniques like ANN approach, Bayesian learning and reinforcement learning.

CLO 3. Able to solve and analyse the problems on ANN, Instance based learning and Reinforcement learning techniques.

CLO 4. To impart the knowledge of clustering and classification Algorithms for predictions and evaluating Hypothesis

Prerequisite: Students should be familiarized about Python installation and setting Python environment

1. Usage and installation of Anaconda should be introduced https://www.anaconda.com/products/individual
2. Should have the knowledge about Probability theory,Statistics theory and linear Algebra.
3. Should have the knowledge of numpy,pandas,scikit-learn and scipy library packages.

Lab Experiments:

PART-A

Program 1:
Aim: Illustrate and Demonstrate the working model and principle of Find-S algorithm.
Program: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.

Program 2:
Aim: Demonstrate the working model and principle of candidate elimination algorithm.
Program: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples

Program 3:
Aim: To construct the Decision tree using the training data sets under supervised learning concept.
Program: Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample

Program 4:
Aim: To understand the working principle of Artificial Neural network with feed forward and feed backward principle.
Program: Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

Program 5:
Aim: Demonstrate the text classifier using Naïve bayes classifier algorithm.
Program: Write a program to implement the naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Program 6
Aim: Demonstrate and Analyse the results sets obtained from Bayesian belief network Principle.
Program:- 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.

Program 7
Aim: Implement and demonstrate the working model of K-means clustering algorithm with Expectation Maximization Concept.
Program: 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.

Program 8:

Aim: Demonstrate and analyse the results of classification based on KNN Algorithm.

Program: Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Program 9

Aim: Understand and analyse the concept of Regression algorithm techniques.

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

Program 10

Aim: Implement and demonstrate classification algorithm using Support vector machine Algorithm.

Program: Implement and demonstrate the working of SVM algorithm for classification.

PART-B

A problem statement for each batch is to be generated in consultation with the co-examiner and student should develop an algorithm, program and execute the Program for the given problem with appropriate outputs

Course Outcomes: At the end of the course the student will be able to:

CO 1. Understand the Importance of different classification and clustering algorithms.

CO 2. Demonstrate the working of various algorithms with respect to training and test data sets.

CO 3. Illustrate and analyze the principles of Instance based and Reinforcement learning techniques.

CO 4. Elicit the importance and Applications of Supervised and unsupervised machine learning.

CO 5. Compare and contrast the Bayes theorem principles and Q learning approach.

Program 1:
Aim: Illustrate and Demonstrate the working model and principle of Find-S algorithm.
Program: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
import csv

# Initialize the hypothesis to the most specific hypothesis
h = ['0', '0', '0', '0', '0', '0']

# Load the dataset from CSV
with open('w-s.csv', 'r') as f:
    reader = csv.reader(f)
    next(reader)  # Skip header
    your_list = list(reader)

# Apply the Find-S algorithm
for x in your_list:
    if len(x) > 0 and x[-1] == "true":
        for j in range(len(x) - 1):  # Iterate over features only
            if x[j] != h[j] and h[j] == '0':
                h[j] = x[j]
            elif x[j] != h[j] and h[j] != '0':
                h[j] = '?'

# Print the maximally specific hypothesis
print("Maximally specific hypothesis is:")
print(h)
```

DATASET:

W-S.CSV:

Sky,temp,Humid,Wind,Water,Forecost,EnjoySpt

Sunny,Warm,Normal,Strong,Warm,same,true

Sunny,Warm,High,Strong,Warm,same,true

Rainy,cold,High,Strong,Warm,Change,false

Sunny,Warm,High,Strong,Cool,Change,true

OUTPUT:

Maximally specific hypothesis is:

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

Program 2:

Aim: Demonstrate the working model and principle of candidate elimination algorithm.

Program: 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

```
import numpy as np
import pandas as pd
data=pd.DataFrame(data=pd.read_csv('tennis.csv'))
concepts=np.array(data.iloc[:,0:-1])
target=np.array(data.iloc[:,-1])

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]='?'
        if target[i]=="no":
            for i 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 in candidate elimination algorithm\n",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")
```

DATASET:

tennis.csv:

sky,airtemp,humidity,wind,water,forecast,enjoysports

'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:

initialization of specific_h and general_h

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

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

steps in candidate elimination algorithm

1

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

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

steps in candidate elimination algorithm

2

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

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

steps in candidate elimination algorithm

6

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

[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], '?', ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', "'same'"]]

steps in candidate elimination algorithm

4

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

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

final specific_h:

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

final general_h:

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

Program 3:

Aim: To construct the Decision tree using the training data sets under supervised learning concept.

Program: 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

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

# Create dataset manually
data = {
    'weather':
    ['sunny', 'sunny', 'cloudy', 'rain', 'rain', 'rain', 'cloudy', 'sunny', 'sunny', 'rain', 'sunny',
     'cloudy', 'cloudy', 'rain'],
    'temperature': ['hot', 'hot', 'hot', 'mild', 'cool', 'cool', 'cool', 'mild', 'cool', 'mild', 'mild', 'mild', 'hot', 'mild']
    ,
    'humidity': ['high', 'high', 'high', 'high', 'normal', 'normal', 'normal', 'high', 'normal', 'normal', 'normal', 'high',
                 'normal', 'high'],
    'wind': ['weak', 'strong', 'weak', 'weak', 'weak', 'strong', 'strong', 'weak', 'weak', 'weak', 'strong', 'strong', 'weak',
             'strong'],
    'class': ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'no']
}

dataset = pd.DataFrame(data)

# Function to calculate entropy
def entropy(target_col):
    elements, counts = np.unique(target_col, return_counts=True)
    entropy = np.sum([(-counts[i] / np.sum(counts)) * np.log2(counts[i] / np.sum(counts)) for i in range(len(elements))])
    return entropy

# Function to calculate information gain
def InfoGain(data, split_attribute_name, target_name="class"):
    total_entropy = entropy(data[target_name])
    vals, counts = np.unique(data[split_attribute_name], return_counts=True)
    Weighted_Entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(data.where(data[split_attribute_name] == vals[i]).dropna(
    Information_Gain = total_entropy - Weighted_Entropy
    return Information_Gain
```

```
# ID3 algorithm to build the decision tree
def ID3(data, originaldata, features, target_attribute_name="class", parent_node_class=None):
    if len(np.unique(data[target_attribute_name])) <= 1:
        return np.unique(data[target_attribute_name])[0]
    elif len(data) == 0:
        return np.unique(originaldata[target_attribute_name])[np.argmax(np.unique(originaldata[target_attribute_name], return_c
    elif len(features) == 0:
        return parent_node_class
    else:
        parent_node_class = np.unique(data[target_attribute_name])[np.argmax(np.unique(data[target_attribute_name], return_coun
        item_values = [InfoGain(data, feature, target_attribute_name) for feature in features]
        best_feature_index = np.argmax(item_values)
        best_feature = features[best_feature_index]
        tree = {best_feature: {}}
        features = [i for i in features if i != best_feature]
        for value in np.unique(data[best_feature]):
            value = value
            sub_data = data.where(data[best_feature] == value).dropna()
            subtree = ID3(sub_data, dataset, features, target_attribute_name, parent_node_class)
            tree[best_feature][value] = subtree
        return tree

# Build the decision tree
tree = ID3(dataset, dataset, dataset.columns[:-1])
print(' \nDisplay Tree\n', tree)
```

DATASET:

Its in program only

OUTPUT:

Display Tree

{'weather': {'cloudy': 'yes', 'rain': {'wind': {'strong': 'no', 'weak': 'yes'}}, 'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}

Program 4:

Aim: To understand the working principle of Artificial Neural network with feed forward and feed backward principle.

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

```
import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)     # X = (hours sleeping, hours studying)
y = np.array(([92], [86], [89]), dtype=float)           # y = score on test

# scale units
X = X/np.amax(X, axis=0)       # maximum of X array
y = y/100                      # max test score is 100
class Neural_Network(object):
    def __init__(self):
                           # Parameters
        self.inputSize = 2
        self.outputSize = 1
        self.hiddenSize = 3
                           # Weights
        self.W1 = np.random.randn(self.inputSize, self.hiddenSize)       # (3x2) weight matrix from input to hidden layer
        self.W2 = np.random.randn(self.hiddenSize, self.outputSize)      # (3x1) weight matrix from hidden to output layer

    def forward(self, X):
                           #forward propagation through our network
        self.z = np.dot(X, self.W1)              # dot product of X (input) and first set of 3x2 weights
        self.z2 = self.sigmoid(self.z)           # activation function
        self.z3 = np.dot(self.z2, self.W2)       # dot product of hidden layer (z2) and second set of 3x1 weights
        o = self.sigmoid(self.z3)                # final activation function
        return o

    def sigmoid(self, s):
        return 1/(1+np.exp(-s))     # activation function

    def sigmoidPrime(self, s):
        return s * (1 - s)          # derivative of sigmoid

    def backward(self, X, y, o):
                                    # backward propgate through the network
```

```
        self.o_error = y - o        # error in output
        self.o_delta = self.o_error*self.sigmoidPrime(o) # applying derivative of sigmoid to
        self.z2_error = self.o_delta.dot(self.W2.T)    # z2 error: how much our hidden layer weights contributed to output erro
        self.z2_delta = self.z2_error*self.sigmoidPrime(self.z2) # applying derivative of sigmoid to z2 error
        self.W1 += X.T.dot(self.z2_delta)       # adjusting first set (input --> hidden) weights
        self.W2 += self.z2.T.dot(self.o_delta)  # adjusting second set (hidden --> output) weights

    def train (self, X, y):
        o = self.forward(X)
        self.backward(X, y, o)
NN = Neural_Network()
for i in range(1000): # trains the NN 1,000 times
    print ("\nInput: \n" + str(X))
    print ("\nActual Output: \n" + str(y))
    print ("\nPredicted Output: \n" + str(NN.forward(X)))

    NN.train(X, y)
```


DATASET:

Its In Program

OUTPUT:

Input:
[[0.66666667 1.   ]
 [0.33333333 0.55555556]
 [1.   0.66666667]]

Actual Output:
[[0.92]
 [0.86]
 [0.89]]

Predicted Output:
[[0.02667726]
 [0.03279832]
 [0.0713175]]

Input:
[[0.66666667 1.   ]
 [0.33333333 0.55555556]
 [1.   0.66666667]]

Actual Output:
[[0.92]
 [0.86]
 [0.89]]

.

.

.

Program 5:

Aim: Demonstrate the text classifier using Naïve bayes classifier algorithm.

Program: Write a program to implement the naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

# Load data from CSV
data = pd.read_csv('tennisdata.csv')
print("THe first 5 values of data is :\n",data.head())
# obtain Train data and Train output
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
# Convert then in numbers
le_Outlook = LabelEncoder()
X.Outlook = le_Outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Wind = LabelEncoder()
X.Wind = le_Wind.fit_transform(X.Wind)

print("\nNow the Train data is :\n",X.head())
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)

classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

DATASET:

Tennisdata.csv:

Outlook,Temperature,Humidity,Wind,PlayTennis

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:

```
THe first 5 values of data is :
     Outlook Temperature Humidity    Wind PlayTennis
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

The First 5 values of train data is
     Outlook Temperature Humidity    Wind
0      Sunny         Hot     High    Weak
1      Sunny         Hot     High  Strong
2   Overcast         Hot     High    Weak
3       Rain        Mild     High    Weak
4       Rain        Cool   Normal    Weak

The first 5 values of Train output is
 0     no
1     no
2    yes
3    yes
4    yes
Name: PlayTennis, dtype: object

Now the Train data is :
    Outlook  Temperature  Humidity  Wind
0        2            1         0     1
1        2            1         0     0
2        0            1         0     1
3        1            2         0     1
4        1            0         1     1

Now the Train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Accuracy is: 1.0
```

Program 6

Aim: Demonstrate and Analyse the results sets obtained from Bayesian belief network Principle.

Program:- 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.

```
import pandas as pd
import numpy as np
import bayespy as bp
import warnings
warnings.filterwarnings('ignore')


heart_disease=pd.read_csv("data7_heart.csv")
print(heart_disease)

#print('Columns in the dataset')
# for col in heart_disease.columns:
#     print(col)

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

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

model.fit(heart_disease, estimator=MaximumLikelihoodEstimator)

from pgmpy.inference import VariableElimination
HeartDisease_infer = VariableElimination(model)

q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'age': 63, 'sex' :0})
print(q)
```

DATA SET:

data7_heart.csv:

age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,heartdisease
63,1,1,145,233,1,2,150,0,2.3,3,0,6,0
67,1,4,160,286,0,2,108,1,1.5,2,3,3,2
……
…….
68,1,4,144,193,1,0,141,0,3.4,2,2,7,2
57,1,4,130,131,0,0,115,1,1.2,2,1,7,3
57,0,2,130,236,0,2,174,0,0,2,1,3,1
38,1,3,138,175,0,0,173,0,0,1,?,3,0

OUTPUT:

```
   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   41    0   2       130   204    0        2      172      0      1.4      1
4   62    0   4       140   268    0        2      160      0      3.6      3
5   60    1   4       130   206    0        2      132      1      2.4      2

   ca  thal  heartdisease
0   0     6             0
1   3     3             2
2   2     7             1
3   0     3             0
4   2     3             3
5   2     7             4
Columns in the dataset
+-----------------+----------------------+
| heartdisease    |   phi(heartdisease) |
+=================+======================+
| heartdisease(0) |               0.1400 |
+-----------------+----------------------+
| heartdisease(1) |               0.2400 |
+-----------------+----------------------+
| heartdisease(2) |               0.2400 |
+-----------------+----------------------+
| heartdisease(3) |               0.2400 |
+-----------------+----------------------+
| heartdisease(4) |               0.1400 |
```

Program 7

Aim: Implement and demonstrate the working model of K-means clustering algorithm with Expectation Maximization Concept.

Program: 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.

```
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()
# print(dataset)
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
# print(X)
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')

# K-PLOT
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')
```

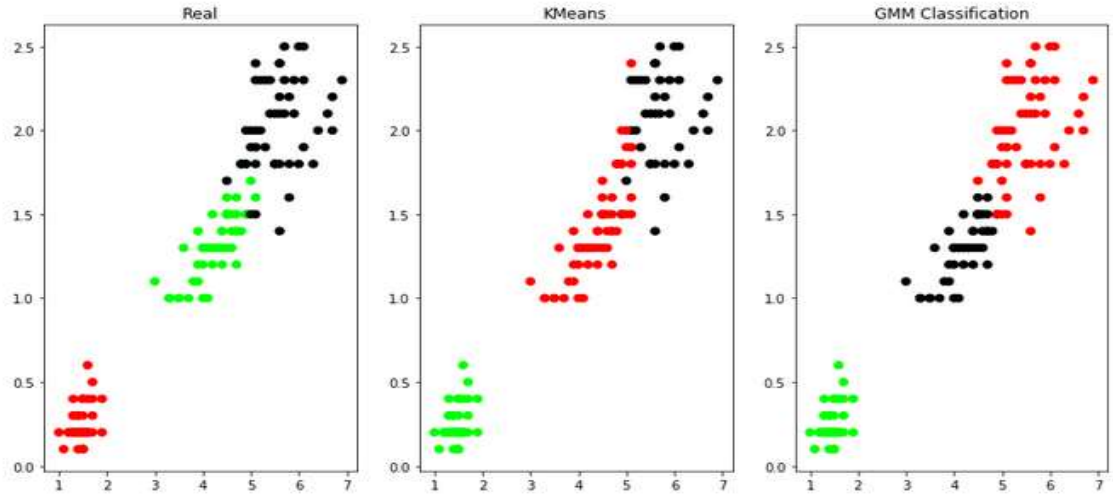
```
# GMM PLOT
scaler=preprocessing.StandardScaler()
scaler.fit(X)
xsa=scaler.transform(X)
xs=pd.DataFrame(xsa,columns=X.columns)
gmm=GaussianMixture(n_components=3)
gmm.fit(xs)
y_cluster_gmm=gmm.predict(xs)
plt.subplot(1,3,3)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)
plt.title('GMM Classification')
```

DATASET:

Loaded from iris repository

OUTPUT:

```
'GMM Classification'
```



Program 8:

Aim: Demonstrate and analyse the results of classification based on KNN Algorithm.

Program: Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

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

# Load the iris dataset
iris = datasets.load_iris()
print("Iris Data set loaded...")

# Split the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.1)
print("Dataset is split into training and testing...")
print("Size of training data and its label:", x_train.shape, y_train.shape)
print("Size of test data and its label:", x_test.shape, y_test.shape)

# Print the labels and their corresponding names
for i in range(len(iris.target_names)):
    print("Label", i, "_", str(iris.target_names[i]))

# Initialize the KNN classifier with k=1
classifier = KNeighborsClassifier(n_neighbors=1)
classifier.fit(x_train, y_train)

# Predict the labels for the test set
y_pred = classifier.predict(x_test)

# Print the results of classification
print("Results of classification using k-nn with k=1")
for r in range(len(x_test)):
    print("Sample:", str(x_test[r]), "Actual label:", str(y_test[r]), "Predicted label:", str(y_pred[r]))
    print("Classification Accuracy:", classifier.score(x_test, y_test))
# Print confusion matrix and classification report
#print('Confusion Matrix:')
#print(confusion_matrix(y_test, y_pred))
```

DATASET:

Loaded from iris repository

OUTPUT:

```
Iris Data set loaded...
Dataset is split into training and testing...
Size of training data and its label: (135, 4) (135,)
Size of test data and its label: (15, 4) (15,)
Label 0 _ setosa
Label 1 _ versicolor
Label 2 _ virginica
Results of classification using k-nn with k=1
Sample: [6.7 3.1 4.4 1.4] Actual label: 1 Predicted label: 1
Classification Accuracy: 0.9333333333333333
Sample: [6.  2.7 5.1 1.6] Actual label: 1 Predicted label: 2
Classification Accuracy: 0.9333333333333333
Sample: [6.9 3.1 4.9 1.5] Actual label: 1 Predicted label: 1
Classification Accuracy: 0.9333333333333333
Sample: [5.9 3.  4.2 1.5] Actual label: 1 Predicted label: 1
Classification Accuracy: 0.9333333333333333
Sample: [5.1 3.8 1.9 0.4] Actual label: 0 Predicted label: 0
Classification Accuracy: 0.9333333333333333
Sample: [7.6 3.  6.6 2.1] Actual label: 2 Predicted label: 2
Classification Accuracy: 0.9333333333333333
Sample: [7.3 2.9 6.3 1.8] Actual label: 2 Predicted label: 2
Classification Accuracy: 0.9333333333333333
Sample: [6.1 2.8 4.7 1.2] Actual label: 1 Predicted label: 1
Classification Accuracy: 0.9333333333333333
Sample: [5.7 2.5 5.  2. ] Actual label: 2 Predicted label: 2
Classification Accuracy: 0.9333333333333333
Sample: [5.  3.2 1.2 0.2] Actual label: 0 Predicted label: 0
Classification Accuracy: 0.9333333333333333
Sample: [6.4 3.1 5.5 1.8] Actual label: 2 Predicted label: 2
Classification Accuracy: 0.9333333333333333
Sample: [5.5 3.5 1.3 0.2] Actual label: 0 Predicted label: 0
Classification Accuracy: 0.9333333333333333
Sample: [4.9 3.1 1.5 0.1] Actual label: 0 Predicted label: 0
```

```
Classification Accuracy: 0.9333333333333333
Sample: [5.8 2.6 4.  1.2] Actual label: 1 Predicted label: 1
Classification Accuracy: 0.9333333333333333
Sample: [4.8 3.4 1.9 0.2] Actual label: 0 Predicted label: 0
Classification Accuracy: 0.9333333333333333
Confusion Matrix:
[[5 0 0]
 [0 5 1]
 [0 0 4]]
Accuracy Metrics:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         5
          1       1.00      0.83      0.91         6
          2       0.80      1.00      0.89         4

avg / total       0.95      0.93      0.93        15
```

Program 9

Aim: Understand and analyse the concept of Regression algorithm techniques.

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

from math import ceil

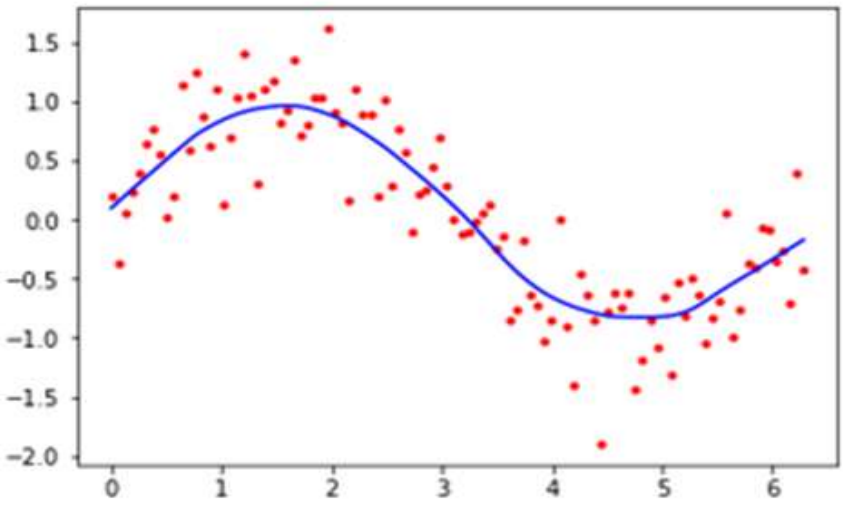
```
from math import ceil
import numpy as np
from scipy import linalg
def lowess(x, y, f, iterations):
    n = len(x)
    r = int(ceil(f * n))
    h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]
    w = np.clip(np.abs((x[:, None] - x[None, :]) / h), 0.0, 1.0)
    w = (1 - w ** 3) ** 3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iterations):
        for i in range(n):
            weights = delta * w[:, i]
            b = np.array([np.sum(weights * y), np.sum(weights * y * x)])
            A = np.array([[np.sum(weights), np.sum(weights * x)],[np.sum(weights * x), np.sum(weights * x * x)]])
            beta = linalg.solve(A, b)
            yest[i] = beta[0] + beta[1] * x[i]

        residuals = y - yest
        s = np.median(np.abs(residuals))
        delta = np.clip(residuals / (6.0 * s), -1, 1)
        delta = (1 - delta ** 2) ** 2

    return yest
import math
n = 100
x = np.linspace(0, 2 * math.pi, n)
y = np.sin(x) + 0.3 * np.random.randn(n)
f =0.25
iterations=3
yest = lowess(x, y, f, iterations)

import matplotlib.pyplot as plt
plt.plot(x,y,"r.")
plt.plot(x,yest,"b-")
```

OUTPUT:



Program 10

Aim: Implement and demonstrate classification algorithm using Support vector machine Algorithm.

Program: Implement and demonstrate the working of SVM algorithm for classification.

```
import numpy as np
import pandas as pd
import  matplotlib.pyplot as plt
df=pd.read_csv('Social_Network_Ads.csv')
df.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)



from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
y_Pred=classifier.predict(X_test)
y_Pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_Pred)
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf',gamma=15,C=7,random_state=0)
classifier.fit(X_train, y_train)
y_Pred=classifier.predict(X_test)
y_Pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_Pred)
from sklearn.svm import SVC
classifier = SVC(kernel = 'poly',degree=4)
classifier.fit(X_train, y_train)
y_Pred=classifier.predict(X_test)
y_Pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_Pred)
```

DATASET:

Social_Network_ads.csv:

User ID,Gender,Age,EstimatedSalary,Purchased
15624510,Male,19,19000,0
15810944,Male,35,20000,0
15668575,Female,26,43000,0
15603246,Female,27,57000,0
15804002,Male,19,76000,0
15728773,Male,27,58000,0
……
…….
15706071,Male,51,23000,1
15654296,Female,50,20000,1
15755018,Male,36,33000,0
15594041,Female,49,36000,1

OUTPUT:

0.9415204678362573