MITRC

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# LAB MANUAL

## B.Tech. VI Semester

## MACHINE LEARNING LAB

## 6CS4-22


**Modern Institute of Technology & Research Centre,**

**Alwar, Rajasthan**

**Affiliated to Bikaner Technical University, Bikaner**

# TABLE OF CONTENT

# LAB INSTRUCTIONS

1. Keep silence and sit properly in the lab.
2. Keep your bag properly inside the lab.
3. Always follow the instruction given by concerned faulty to perform the assigned experiment.
4. Do not turn on the PCs without permission.
5. Do not switch off the power supply of the PCs directly, first shut down the PCs then switch off power supply.
6. Every student is responsible for any damage to the PCs or its accessories which is assigned for lab work.
7. Installation or un-installation of any software is strictly prohibited.
8. Always bring your lab file and the task assigned to you must be completed.
9. Experiment performed by you should be positively checked in next turn after that faculty may not check your work.
10. Please mention your roll number, name, node number and signature in lab register.
11. Do not go to assist any student.

# BTU SYLLABUS

| S.No | Content |
|---|---|
| 1 | Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. |
| 2 | For a given set of training data examples, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples. |
| 3 | Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample |
| 4 | Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets |
| 5 | Write a program to implement the naive Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets. |
| 6 | Assuming a set of documents that need to be classified, use the naive Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set. |
| 7 | Write a program to construct aBayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API. |
| 8 | Apply EM algorithm to cluster a set of data. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program. |
| 9 | Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem. |
| 10 | Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set and draw graphs. |

# LAB INTRODUCTION

This lab is intended for the third year students of engineering branches in the subject of Machine Learning. This manual typically contains practical/Lab Sessions related ML covering various aspects related to the subject to enhance understanding.

The programs are implemented in python programming language and involve use of packages like numpy, pandas, matplotlib, scikit-learn.

# Program No. 1

**Objective:** Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

**Program** Python implementation of FIND-S algorithm

```
import csv
a = []

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

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

num_attribute = len(a[0])-1

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

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

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

Listing 1: Find-S program



## Output

```
The Given Training Data Set
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']
The total number of training instances are : 4
The initial hypothesis is :
```

```
['0', '0', '0', '0', '0', '0']
The hypothesis for the training instance 1 is :
['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
The hypothesis for the training instance 2 is :
['sunny', 'warm', '?', 'strong', 'warm', 'same']
The hypothesis for the training instance 3 is :
['sunny', 'warm', '?', 'strong', 'warm', 'same']
The hypothesis for the training instance 4 is :
['sunny', 'warm', '?', 'strong', '?', '?']
The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
```

## Viva Voce

1. What do you mean by Concept learning?
2. What do you mean by Inductive logic?
3. What do you understand by Hypothesis space?
4. Can Concept learning be used as searching algorithm?
5. Explain the Find-S Algorithm.

## Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 2

**Objective:** 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** Python implementation of Candidate-Elimination algorithm

```
import numpy as np
import pandas as pd

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

    for i, h in enumerate(concepts):
        print("For Loop Starts")
        if target[i] == "yes":
            print("If instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

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

        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
        print("\n")
        print("\n")

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

```
43
44 s_final, g_final = learn(concepts, target)
45
46 print("Final Specific_h:", s_final, sep="\n")
47 print("Final General_h:", g_final, sep="\n")
```

Listing 2: Candidate-Elimination program

## Output

```
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'],
['?', 'warm', '?', '?', '?', '?']]
```

## Viva Voce

1. What do you mean by consistent hypothesis?
2. What do you mean by version space?
3. Explain Candidate-Elimination ALgorithm.?
4. How Candidate-Elimination is different from Find-S algorithm.

## Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 3

**Objective:** 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** Python implementation of decision tree based on ID3 algorithm

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

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

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

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

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

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

    counts=[0,0]
```

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

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

def compute_gain(data,col):
    attr,dic = subtables(data,col,delete=False)

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

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

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

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


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

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

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

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

```

```
100
101 def classify(node,x_test,features):
102     if node.answer!="":
103         print(node.answer)
104         return
105     pos=features.index(node.attribute)
106     for value, n in node.children:
107         if x_test[pos]==value:
108             classify(n,x_test,features)
109
110 '''Main program'''
111 dataset,features=load_csv("id3.csv")
112 node1=build_tree(dataset,features)
113
114 print("The decision tree for the dataset using ID3 algorithm is")
115 print_tree(node1,0)
116 testdata,features=load_csv("id3_test.csv")
117
118 for xtest in testdata:
119     print("The test instance:",xtest)
120     print("The label for test instance:",end="    ")
121     classify(node1,xtest,features)
```

Listing 3: Decision tree using ID3 algorithm

## Output

The decision tree for the dataset using ID3 algorithm is

```
Outlook
     rain
          Wind
               strong
                    no
               weak
                    yes
     overcast
          yes
     sunny
          Humidity
               normal
                    yes
               high
                    no
The test instance: ['rain', 'cool', 'normal', 'strong']
The label for test instance: no
```

## Viva Voce

1. How you decide the root node of decision tree?

2. What is role of entropy in decision tree?
3. what is pruning of decision tree?

# Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 4

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

**Program** ANN using backpropagation algorithm implemented in python language

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float) # two inputs [sleep ,study]
y = np.array(([92], [86], [89]), dtype=float) # one output [Expected % in Exams]
X = X/np.amax(X,axis=0) # maximum of X array longitudinally
y = y/100

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

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

#Variable initialization
epoch=5000  #Setting training iterations
lr=0.1    #Setting learning rate
inputlayer_neurons = 2    #number of features in data set
hiddenlayer_neurons = 3   #number of hidden layers neurons
output_neurons = 1    #number of neurons at output layer

#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons)) # weight of the link from input node to hidden node
bh=np.random.uniform(size=(1,hiddenlayer_neurons)) # bias of the link from input node to hidden node
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons)) # weight of the link from hidden node to output node
bout=np.random.uniform(size=(1,output_neurons)) #bias of the link from hidden node to output node


#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):

#Forward Propogation
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp= outinp1+ bout
    output = sigmoid(outinp)

```

```
#Backpropagation
    EO = y-output
    outgrad = derivatives_sigmoid(output)
    d_output = EO* outgrad
    EH = d_output.dot(wout.T)

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

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

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

Listing 4: ANN backpropogation algorithm implementation

# Output

# Viva Voce

1. What is pereceptron?
2. how multi-layer ANN is better than single layer ANN?
3. what is Back-Propogation Algorithm?

# Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 5

**Objective:** 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** Naive Bayesian classifier implemented in python

```
import csv
import random
import math

def loadcsv(filename):
  lines = csv.reader(open(filename, "r"));
  dataset = list(lines)
  for i in range(len(dataset)):
       #converting strings into numbers for processing
    dataset[i] = [float(x) for x in dataset[i]]

  return dataset

def splitdataset(dataset, splitratio):
    #67% training size
  trainsize = int(len(dataset) * splitratio);
  trainset = []
  copy = list(dataset);
  while len(trainset) < trainsize:
#generate indices for the dataset list randomly to pick ele for
    training data
    index = random.randrange(len(copy));
    trainset.append(copy.pop(index))
  return [trainset, copy]

def separatebyclass(dataset):
  separated = {} #dictionary of classes 1 and 0
#creates a dictionary of classes 1 and 0 where the values are
#the instances belonging to each class
  for i in range(len(dataset)):
    vector = dataset[i]
    if (vector[-1] not in separated):
      separated[vector[-1]] = []
    separated[vector[-1]].append(vector)
  return separated

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

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

```

```
def summarize(dataset): #creates a dictionary of classes
  summaries = [(mean(attribute), stdev(attribute)) for attribute in zip
    (*dataset)];
  del summaries[-1] #excluding labels +ve or -ve
  return summaries

def summarizebyclass(dataset):
  separated = separatebyclass(dataset);
    #print(separated)
  summaries = {}
  for classvalue, instances in separated.items():
#for key,value in dic.items()
#summaries is a dic of tuples(mean,std) for each class value
    summaries[classvalue] = summarize(instances) #summarize is used to
    cal to mean and std
  return summaries

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

def calculateclassprobabilities(summaries, inputvector):
  probabilities = {} # probabilities contains the all prob of all class
    of test data
  for classvalue, classsummaries in summaries.items():#class and
    attribute information as mean and sd
    probabilities[classvalue] = 1
    for i in range(len(classsummaries)):
      mean, stdev = classsummaries[i] #take mean and sd of every
    attribute for class 0 and 1 seperaely
      x = inputvector[i] #testvector's first attribute
      probabilities[classvalue] *= calculateprobability(x, mean, stdev)
    ;#use normal dist
  return probabilities

def predict(summaries, inputvector): #training and test data is passed
  probabilities = calculateclassprobabilities(summaries, inputvector)
  bestLabel, bestProb = None, -1
  for classvalue, probability in probabilities.items():#assigns that
    class which has he highest prob
    if bestLabel is None or probability > bestProb:
      bestProb = probability
      bestLabel = classvalue
  return bestLabel

def getpredictions(summaries, testset):
  predictions = []
  for i in range(len(testset)):
    result = predict(summaries, testset[i])
    predictions.append(result)
  return predictions

def getaccuracy(testset, predictions):
  correct = 0
  for i in range(len(testset)):
```

```
92      if testset[i][-1] == predictions[i]:
93        correct += 1
94    return (correct/float(len(testset))) * 100.0
95
96  def main():
97    filename = 'naivedata.csv'
98    splitratio = 0.67
99    dataset = loadcsv(filename);
100
101   trainingset, testset = splitdataset(dataset, splitratio)
102   print('Split {0} rows into train={1} and test={2} rows'.format(len(
        dataset), len(trainingset), len(testset)))
103   # prepare model
104   summaries = summarizebyclass(trainingset);
105   #print(summaries)
106     # test model
107   predictions = getpredictions(summaries, testset) #find the
        predictions of test data with the training data
108   accuracy = getaccuracy(testset, predictions)
109   print('Accuracy of the classifier is : {0}%'.format(accuracy))
110
111 main()
```

Listing 5: Bayesian Classifier implementation

## Output

```
Split 768 rows into train=514 and test=254 rows
Accuracy of the classifier is : 71.65354330708661%
```

## Viva Voce

1. What do you understand by classification Algorithm in machine learning?
2. What do you mean by conditional probability?
3. Definte Bayes Theorem?
4. Explain how Bayes theorem is used for classification?

## Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 6

**Objective:** Assuming a set of documents that need to be classified, use the naive Bayesian Classifier model to perform this task.Calculate the accuracy, precision, and recall for your data set.

**Program** Naive Bayesian classifier implemented using python Scikit-learn package.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

msg=pd.read_csv('naivetext.csv',names=['message','label'])

print('The dimensions of the dataset',msg.shape)

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

#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)


#output the words or Tokens in the text documents
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())

# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,
    predicted))
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,
    predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

Listing 6: Bayesian Classifier

# 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

Name: labelnum, dtype: int64
```

```
The total number of Training Data: (13,)
The total number of Test Data: (5,)

The words or Tokens in the text documents

['about', 'am', 'amazing', 'an', 'and', 'awesome', 'beers',
 'best', 'can', 'deal', 'do', 'enemy', 'feel',
 'fun', 'good', 'great', 'have', 'he', 'holiday', 'house',
 'is', 'like', 'love', 'my', 'not', 'of', 'place',
'restaurant', 'sandwich', 'sick', 'sworn', 'these', 'this',
'tired', 'to', 'today', 'tomorrow', 'very',
'view', 'we', 'went', 'what', 'will', 'with', 'work']

Accuracy of the classifier is 0.8

Confusion matrix
[[2 1]
[0 2]]

The value of Precision 0.6666666666666666
The value of Recall 1.0
```



## Viva Voce

1. What do you understand by classification Algorithm in machine learning?
2. What do you mean by conditional probability?
3. Definte Bayes Theorem?
4. Explain how Bayes theorem is used for classification?

## Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 7

**Objective:** 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.

**Program** Bayesian network constructed using python pgmpy package used for making network.

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

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

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

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

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

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

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

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

Listing 7: Bayesian Network

# Output

```
Few examples from the dataset are given below
     age  sex cp  trestbps   ...slope  ca  thal  heartdisease
0    63    1   1      145       ...   3   0     6             0
1    67    1   4      160       ...   2   3     3             2
2    67    1   4      120       ...   2   2     7             1
3    37    1   3      130       ...   3   0     3             0
4    41    0   2      130       ...   1   0     3             0

[5 rows x 14 columns]

Learning CPD using Maximum likelihood estimators

Inferencing with Bayesian Network:


2. Probability of HeartDisease given cholesterol=100
```

| heartdisease | phi(heartdisease) |
|---|---|
| heartdisease_0 | 0.5400 |
| heartdisease_1 | 0.1533 |
| heartdisease_2 | 0.1303 |
| heartdisease_3 | 0.1259 |
| heartdisease_4 | 0.0506 |



# Viva Voce

1. How you construct Bayes Network?
2. What are the advantages of Bayes Network ?

# Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 8

**Objective:** 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.

**Program** K-means algorithm implemented using python scikit-learn package.

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

iris = datasets.load_iris()

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

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

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

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

# View the results

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

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

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

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

# View the results
```

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

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

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

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

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

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

y_cluster_gmm = gmm.predict(xs)
#y_cluster_gmm

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

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

Listing 8: Clustering



## Viva Voce

1. Is clustering unsupervised or supervised learning ?
2. Name few important clustering algorithm.

3. What are advantages of k-means clustering?

# Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 9

**Objective:** Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

**Program** KNN algorithm implemented using python scikit-learn package.

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

iris=datasets.load_iris()

x = iris.data
y = iris.target

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

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

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

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

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

Listing 9: KNN implemenation



## Output

```
sepal-length sepal-width petal-length petal-width
[[5.1 3.5 1.4 0.2]
[4.9 3. 1.4 0.2]
[4.7 3.2 1.3 0.2]
[4.6 3.1 1.5 0.2]
[5. 3.6 1.4 0.2]
. . . . .
. . . . .
[6.2 3.4 5.4 2.3]
```

```
[5.9 3. 5.1 1.8]]
class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica
[0 0 0 ......0 0 1 1 1 ........1 1 2 2 2 ........ 2 2]

Confusion Matrix[[20 0 0]
[ 0 10 0]
[ 0 1 14]]

Accuracy Metrics
Precision recall f1-score support
0 1.00 1.00 1.00 20
1 0.91 1.00 0.95 10
2 1.00 0.93 0.97 15
avg / total 0.98 0.98 0.98 45
```

## Viva Voce

1. What are features of iris-data?
2. Explain K-nearest neighbour algorithm?
3. K-nearest neighbour is regression or classification?

## Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.

# Program No. 10

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

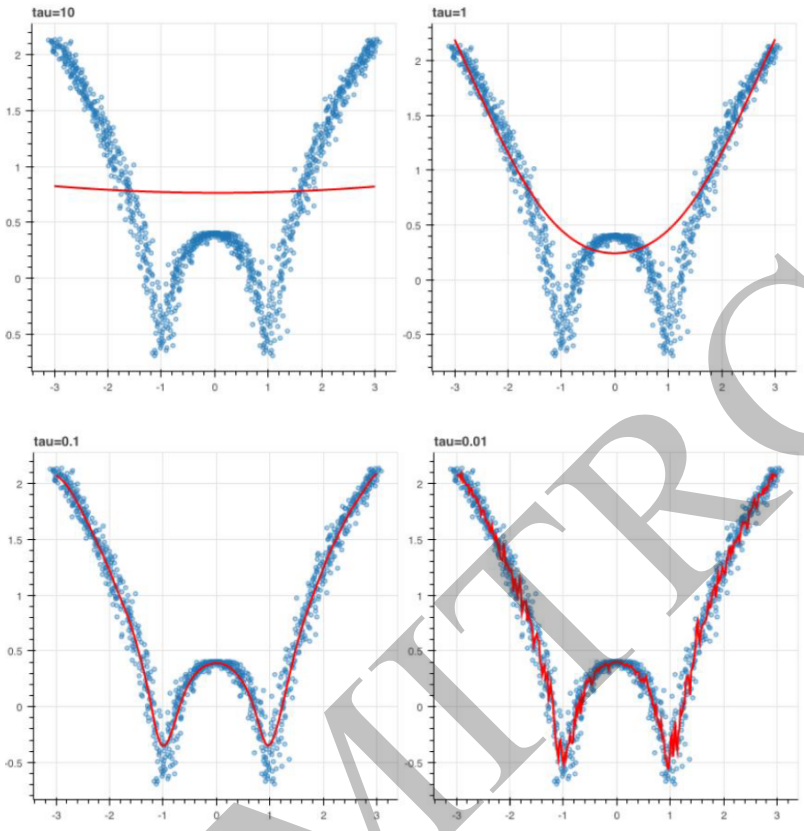
**Program** Non-parametric regression implemented using python language.

```
import numpy as np
from bokeh.plotting import figure, show, output_notebook
from bokeh.layouts import gridplot
from bokeh.io import push_notebook

def local_regression(x0, X, Y, tau):# add bias term
    x0 = np.r_[1, x0] # Add one to avoid the loss in information
    X = np.c_[np.ones(len(X)), X]
    # fit model: normal equations with kernel
    xw = X.T * radial_kernel(x0, X, tau) # XTranspose * W
    beta = np.linalg.pinv(xw * X) * xw * Y #@ Matrix Multiplication or Dot Product
    # predict value
    return x0 * beta # @ Matrix Multiplication or Dot Product for prediction


def radial_kernel(x0, X, tau):
    return np.exp(np.sum((X - x0) ** 2, axis=1) / (-2 * tau * tau))
# Weight or Radial Kernal Bias Function

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

domain = np.linspace(-3, 3, num=300)
print(" Xo Domain Space(10 Samples) :\n",domain[1:10])

def plot_lwr(tau):
    # prediction through regression
    prediction = [local_regression(x0, X, Y, tau) for x0 in domain]
    plot = figure(plot_width=400, plot_height=400)
    plot.title.text='tau=%g' % tau
    plot.scatter(X, Y, alpha=.3)
    plot.line(domain, prediction, line_width=2, color='red')
    return plot

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

```
44  [plot_lwr(0.1), plot_lwr(0.01)]]))
```

Listing 10: Non-parametric regression

# Output






# Viva Voce

1. What is difference between regression and classification?
2. Is regression supervised learning?
3. Name few important regression algorithm.

# Reference

1. Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
2. Liu, Yuxi Hayden. *Python Machine Learning By Example: Implement machine learning algorithms and techniques to build intelligent systems.* Packt Publishing Ltd, 2019.