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

import csv

a = []

with open('/home/cit/Downloads/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)

The hypothesis for the training instance 5 is :

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

The Maximally specific hypothesis for the training instance is

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

**2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples**

import csv

with open("trainingexamples.csv") as f:

csv\_file = csv.reader(f)

data = list(csv\_file)

specific = data[1][:-1]

general = [['?' for i in range(len(specific))] for j in range(len(specific))]

for i in data:

if i[-1] == "Yes":

for j in range(len(specific)):

if i[j] != specific[j]:

specific[j] = "?"

general[j][j] = "?"

elif i[-1] == "No":

for j in range(len(specific)):

if i[j] != specific[j]:

general[j][j] = specific[j]

else:

general[j][j] = "?"

print("\nStep " + str(data.index(i)+1) + " of Candidate Elimination Algorithm")

print(specific)

print(general)

gh = [] # gh = general Hypothesis

for i in general:

for j in i:

if j != '?':

gh.append(i)

break

print("\nFinal Specific hypothesis:\n", specific)

print("\nFinal General hypothesis:\n", gh)

#### 4.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)

y = np.array(([92], [86], [89]), dtype=float)

X = X/np.amax(X,axis=0) # maximum of X array longitudinally

y = y/100

#Sigmoid Function

def sigmoid (x):

return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function

def derivatives\_sigmoid(x):

return x \* (1 - x)

#Variable initialization

epoch=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))

bh=np.random.uniform(size=(1,hiddenlayer\_neurons))

wout=np.random.uniform(size=(hiddenlayer\_neurons,output\_neurons))

bout=np.random.uniform(size=(1,output\_neurons))

#draws a random range of numbers uniformly of dim x\*y

for i in range(epoch):

#Forward Propogation

hinp1=np.dot(X,wh)

hinp=hinp1 + bh

hlayer\_act = sigmoid(hinp)

outinp1=np.dot(hlayer\_act,wout)

outinp= outinp1+ bout

output = sigmoid(outinp)

#Backpropagation

EO = y-output

outgrad = derivatives\_sigmoid(output)

d\_output = EO\* outgrad

EH = d\_output.dot(wout.T)

#how much hidden layer wts 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)

#### 5.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 few test data sets.

#### # import necessary libraries

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 the train data is\n", X.head())

y = data.iloc[:, -1]

print("\nThe First 5 values of train output is\n", y.head())

# convert them 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\_Windy = LabelEncoder()

X.Windy = le\_Windy.fit\_transform(X.Windy)

print("\nNow the Train output 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))