PROGRAM-03

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 numpy as np
import pandas as pd
data = pd.read_csv('finds.csv')
def train(concepts, target):
    for i, val in enumerate(target):
        if val == "Yes":
            specific_h = concepts[i]
            break

    for i,h in enumerate(concepts):
        if target[i] == "Yes":
            for x in range(len(specific_h)):
                if h[x] == specific_h[x]:
                    pass
                else:
                    specific_h[x] = "?"
                    return specific_h

concepts = np.array(data.iloc[:,0:-1]) #slicing rows and column, : means
begining to end of row
target = np.array(data.iloc[:,-1])

print(train(concepts,target))
```

finds.csv (csv file for Program-03)

Sky	Airtemp	Humidity	Wind	Water	Forecast	WaterSport
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 FOR PROGRAM-03

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

o/p : ['Sunny' 'warm' '?' 'Strong' '?' '?']

PROGRAM-04

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 numpy as np
import pandas as pd
# Loading Data from a CSV File
data = pd.DataFrame(data=pd.read_csv('finds.csv'))
# Separating concept features from Target
concepts = np.array(data.iloc[:,0:-1])
# Isolating target into a separate DataFrame
target = np.array(data.iloc[:,-1])
def learn(concepts, target):
    '''
    learn() function implements the learning method of the Candidate
elimination algorithm.
    Arguments:
    concepts - a data frame with all the features
    target - a data frame with corresponding output values

    # Initialise S0 with the first instance from concepts
    # .copy() makes sure a new list is created instead of just pointing
to the same memory location '''
    specific_h = concepts[0].copy()
    # Initialises G0 using list comprehension
    # Creates as many lists inside as there are arguments,
    # that which later will be replaced with actual parameters
    # G0 = [['?', '?', '?', '?', '?', '?'],
    #       ['?', '?', '?', '?', '?', '?'],
    #       ['?', '?', '?', '?', '?', '?'],
    #       ['?', '?', '?', '?', '?', '?'],
    #       ['?', '?', '?', '?', '?', '?'],
    #       ['?', '?', '?', '?', '?', '?']]
    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
        # The learning iterations
    for i, h in enumerate(concepts):
        # Checking if the hypothesis has a positive target
        if target[i] == "Yes":
            for x in range(len(specific_h)):

                # Change values in S & G only if values change
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'

        # Checking if the hypothesis has a positive target
        if target[i] == "No":
            for x in range(len(specific_h)):
                # For negative hyposthesis change values only  in G
                if h[x] != specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
    # find indices where we have empty rows, meaning those that are
unchanged
    indices = [i for i,val in enumerate(general_h) if val == ['?', '?',
'?', '?', '?', '?']]
    for i in indices:
```

```
        # remove those rows from general_h
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    # Return final values
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final S:", s_final, sep="\n")
print("Final G:", g_final, sep="\n")
data.head()
```

finds.csv **(csv file for Program-04)**

Sky	Airtemp	Humidity	Wind	Water	Forecast	WaterSport
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 FOR PRPGRAM-04

Final S:
['Sunny' 'Warm' '?' 'Strong' '?' '?']

Final G:
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]

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

```
import numpy as np

import matplotlib as m


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=7000

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

Output:

Input:

[[0.66666667 1.   ]
[0.33333333 0.55555556]
[1.   0.66666667]]

Actual Output:

[[0.92]
[0.86]
[0.89]]

Predicted Output:

[[0.89396484]
[0.8797435]
[0.89613221]]

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

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

dataset=datasets.load_diabetes()
model=GaussianNB()
model.fit(dataset.data,dataset.target)
expected=dataset.target
predicted=model.predict(dataset.data)
print(metrics.confusion_matrix(expected,predicted))
print(metrics.accuracy_score(expected,predicted))
```

Output:

```
[[1 0 0 ... 0 0 0]
[0 1 0 ... 0 0 0]
[0 0 1 ... 0 0 0]
...
[0 0 0 ... 1 0 0]
[0 0 0 ... 0 1 0]
[0 0 0 ... 0 0 1]]
0.45248868778280543
```

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.

```
from pomegranate import*
asia= DiscreteDistribution({'True':0.5,'False':0.5})
tuberculosis=ConditionalProbabilityTable(
    [['True','True',0.2],
    ['True','False',0.8],
    ['False','True',0.01],
    ['False','False',0.99]],[asia])
smoking=DiscreteDistribution({'True':0.5,'False':0.5})
lung=ConditionalProbabilityTable(
    [['True','True',0.75],
    ['True','False',0.25],
    ['False','True',0.02],
    ['False','False',0.98]],[smoking])
bronchitis=ConditionalProbabilityTable(
    [['True','True',0.92],
    ['True','False',0.08],
    ['False','True',0.03],
    ['False','False',0.97]],[smoking])
tuberculosis_or_cancer=ConditionalProbabilityTable(
    [['True','True','True',1.0],
    ['True','True','False',0.0],
```

```
            ['True','False','True',1.0],
            ['True','False','False',0.0],
            ['False','True','True',1.0],
            ['False','True','False',0.0],
            ['False','False','True',0.0],
            ['False','False','False',1.0]],[tuberculosis,lung])
    xray=ConditionalProbabilityTable(
        [['True','True',0.885],
        ['True','False',0.115],
        ['False','True',0.04],
        ['False','False',0.96]],[tuberculosis_or_cancer])
    dyspnea=ConditionalProbabilityTable(
        [['True','True','True',0.96],
        ['True','True','False',0.04],
        ['True','False','True',0.89],
        ['True','False','False',0.11],
        ['False','True','True',0.96],
        ['False','True','False',0.04],
        ['False','Flase','True',0.89],
        ['False','False','False',0.11]],[tuberculosis_or_cancer,bronchitis])
s0=State(asia,name='asia')
s1=State(tuberculosis,name="tuberculosis")
s2=State(smoking,name="smoker")
network=BayesianNetwork("asia")
network.add_nodes(s0,s1,s2)
network.add_edge(s0,s1)
network.add_edge(s1,s2)
network.bake()
print(network.predict_proba({'tuberculosis':'True'}))
```

Output:


```
[{
"class" :"Distribution",
"dtype" :"str",
"name" :"DiscreteDistribution",
"parameters" :[
  {
    "True" :0.9523809523809521,
    "False" :0.04761904761904782
  }
],
"frozen" :false
}
'True'
{
"class" :"Distribution",
"dtype" :"str",
"name" :"DiscreteDistribution",
"parameters" :[
  {
    "True" :0.5,
    "False" :0.5
  }
],
"frozen" :false
}]
```

8. 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.metrics import confusion_matrix,classification_report

from sklearn import datasets

iris=datasets.load_iris()

iris_data=iris.data

iris_labels=iris.target

print(iris_data)

x_train,X_test,Y_train,Y_test=train_test_split(iris_data,iris_labels,test_size=0.20)

classifier=KNeighborsClassifier(n_neighbors=5)

classifier.fit(x_train,Y_train)

y_prd=classifier.predict(X_test)

print(confusion_matrix(Y_test,y_prd))

print(classification_report(Y_test,y_prd))
```

Output:

```
[[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]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
```

```
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]
[5.1 3.5 1.4 0.3]
[5.7 3.8 1.7 0.3]
[5.1 3.8 1.5 0.3]
[5.4 3.4 1.7 0.2]
[5.1 3.7 1.5 0.4]
[4.6 3.6 1.  0.2]
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
```

35

```
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
```

```
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
[[12  0  0]
 [ 0  5  0]
 [ 0  0 13]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00        13

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

09. 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 numpy import *
import operator
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy.linalg
from scipy.stats.stats import pearsonr

def kernel(point,xmat,k):
  m,n=shape(xmat)
  weights=mat(eye((m)))
  for j in range(m):
    diff=point-X[j]
    weights[j,j]=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=shape(xmat)
  ypred=zeros(m)
  for i in range(m):
    ypred [i]=xmat[i]*localWeight(xmat[i],xmat,ymat,k)
  return ypred
```

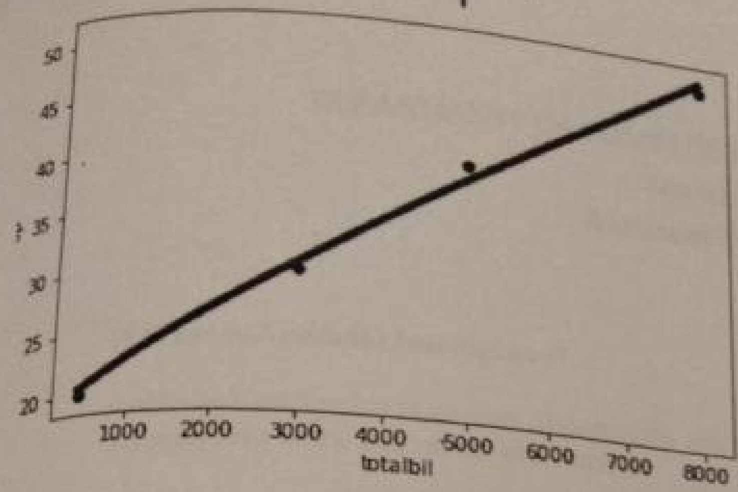
```
data=pd.read_csv('tips.csv')
bill=array(data.bill)
tip=array(data.tip)
mbill=mat(bill)
mtip=mat(tip)
m=shape(mbill)[1]
one=mat(ones(m))
X=hstack((one.T,mbill.T))

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('totalbil')
plt.ylabel('tip')
plt.show();
```

tips.csv

```
bill,tip
3000,30
400,20
5000,40
8000,50
```

OUTPUT

OUTPUT


50
45
40
35
30
25
20
1000
2000
3000
4000
5000
6000
7000
8000
totalbil

40 70

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manu Y M
Assistant Professor

1) What is Artificial Intelligence?

Artificial Intelligence is an area of computer science that emphasizes the creation of intelligent machine that work and reacts like humans.

2) What is an artificial intelligence Neural Networks?

Artificial intelligence Neural Networks can model mathematically the way biological brain works, allowing the machine to think and learn the same way the humans do- making them capable of recognizing things like speech, objects and animals like we do.

3) What are the various areas where AI (Artificial Intelligence) can be used?

Artificial Intelligence can be used in many areas like Computing, Speech recognition, Bio-informatics, Humanoid robot, Computer software, Space and Aeronautics's etc.

4) Which is not commonly used programming language for AI?

Perl language is not commonly used programming language for AI

5) What is Prolog in AI?

In AI, Prolog is a programming language based on logic.

6) Give an explanation on the difference between strong AI and weak AI?

Strong AI makes strong claims that computers can be made to think on a level equal to humans while weak AI simply predicts that some features that are resembling to human intelligence can be incorporated to computer to make it more useful tools.

7) Mention the difference between statistical AI and Classical AI ?

Statistical AI is more concerned with "inductive" thought like given a set of pattern, induce the trend etc. While, classical AI, on the other hand, is more concerned with " deductive" thought given as a set of constraints, deduce a conclusion etc.

8) Which search method takes less memory?

The "depth first search" method takes less memory.

- Expert Systems
- Fuzzy Logic Systems
- Natural Language Processing

18)What is AO* Algorithm

AO* Algorithm basically based on problem decomposition (Breakdown problem into small pieces)

When a problem can be divided into a set of sub problems, where each sub problem can be solved separately and a combination of these will be a solution, **AND-OR graphs** or **AND - OR trees** are used for representing the solution.

The decomposition of the problem or problem reduction generates AND arcs.

COMPUTER SCIENCE & ENGINEERING
Manu Y M
Assistant Professor

ML Viva Questions

1. What is machine learning?
2. Define supervised learning
3. Define unsupervised learning
4. Define semi supervised learning
5. Define reinforcement learning
6. What do you mean by hypotheses
7. What is classification
8. What is clustering
9. Define precision, accuracy and recall
10. Define entropy
11. Define regression
12. How Knn is different from k-means clustering
13. What is concept learning
14. Define specific boundary and general boundary
15. Define target function
16. Define decision tree
17. What is ANN

18. Explain gradient descent approximation
19. State Bayes theorem
20. Define Bayesian belief networks
21. Differentiate hard and soft clustering
22. Define variance
23. What is inductive machine learning
24. Why K nearest neighbour algorithm is lazy learning algorithm
25. Why naïve Bayes is naïve
26. Mention classification algorithms
27. Define pruning
28. Differentiate Clustering and classification
29. Mention clustering algorithms
30. Define Bias