Date. 29/9/20

Expt. No. 1 Page No. 1

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

```
import random
import csv
attributes = [['sunny','Rainy'],['warm','cold'],['normal',
  'high'],['strong','weak'],['warm','cool'],['same',
        'change']]
  print(attributes)
  num_attributes = len(attributes)
  print(num_attributes)
  print("\n the most general hypothesis : ['?','?','?',
              '?','?','?']\n")
  print("\n the most specific hypothesis : ['0','0','0',
                '0','0','0']\n")
  a=[]
  print("\n the given training dataset")
  with open('C:\\Users\\Admin\\Desktop\\AIML\\data.csv',
          'r') as csvfile:
     reader = csv.reader(csvfile)
      for row in reader:
           a.append(row)
          print(row)
  print("\n the initial value of hypothesis:")
  h= ['0']*num_attributes
  print(h)
  for j in range(0, numattributes):
```

Teacher's Signature :

Date....................

Expt. No......1............ Page No.....2..............

```
b[j]= a[i][j]
for i in range(1, len(a)):
    if (a[i][num_attributes] == 'Yes'):
        for j in range(num_attributes):
            if(b[j]=='0' or b[j] == a[i][j]):
                b[j] = a[i][j]
            else:
                b[j] = '?'
    print("In for training examples: {0} the
    hypothesis") format (i+1), b)
```

Teacher's Signature : ....

Dataset :

Sl. No	sky	Airtemp	Humidity	wind	water	forecast	enjoy sport
1	sunny	warm	normal	strong	warm	same.	Yes
2	sunny	warm	high	strong	warm	same	Yes
3	Rainy	cold.	high.	strong.	warm	change.	No
4.	sunny	warm	high	strong	cool	change	Yes.

output:-

[['sunny', 'Rainy'] ['warm', 'cold'], ['normal', 'high'],
['strong', 'weak'], ['warm', 'cool'], ['same', 'change']]

b.

the most general hypothesis : ['?', '?', '?', '?', '?', '?']

the most specific hypothesis : ['o', 'o', 'o', 'o', 'o', 'o']

The given training dataset
['sky', 'AirTemp', 'Humidity', 'wind', 'water', 'forecast',
'Enjoysport']

['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'Yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['Rainy', 'cold', 'High', 'strong', 'warm', 'change', 'yes']
The initial hypothesis is :
['o', 'o', 'o', 'o', 'o', 'o']

for training example : {0} the hypothesis is
['sunny', 'warm', '?', 'strong', '?', '?']

2. for a given set of set of training data example stored in a .csv file, implement and demonstrate the candidate-elimination algorithms to output a description of the set of all hypothesis consistent with the training example.

```
import csv
with open('c:\\users\\Admin\\Desktop\\4Mb1\\albin.csv')
as f:
csv_file = csv.reader(f)
data = list(csv_file)
print(data)
 s= data[1][:-1]
 g= [['?' for i in range(len(s))] for j in range(
    len(s))]
 for i in data:
     if i[-1]=="Yes":
        for j in range(len(s)):
            If i[j]!= s[j]:
               s[j] = '?'
               g[j][j]='?'
   elif i[-1]=="No":
      for j in range(len(s)):
         if i[j]!= s[j]:
            g[j][j] = s[j].
      else:
            g[j][j] = "?"
  print("\n steps of candidate elimination algori-
  thms", data.index(i)+1)
```

Teacher's Signature : ....

Date....................

Expt. No. 2 Page No. 4

```
Prohnt(s)
 print(g)
 gh=[]
  for i in g:
      for j in i:
         if j!='?':
            gh.append(i)
            break
print("\n final specific hypothesis:\n",s)
print("\n final general hypothesis:\n",gh)
```

Teacher's Signature : ....

Date..................

Expt. No.................. Page No..................

Dataset.

Sunny	warm	normal	strong	warm	same	Yes
Sunny	warm	high	strong	warm	same	Yes
Rainy	cold	high	strong	warm	change	Yes
Sunny	warm	high	strong	warm	change	Yes

Output:

steps of candidate elimination algorithm 1
['sunny','warm','?','strong','?','?']
[['sunny','?','?','?','?','?'], ['?','warm','?','?','?','?'],
['?','?','?','?','?','?'], ['?','?','?','?','?','?'], ['?','?','?','?','?','?'],
'?','?','?','?','?','?']]

steps of candidate elimination algorithm 2
['sunny','warm','?','strong','?','?']
[['sunny','?','?','?','?','?'], ['?','warm','?','?','?','?'],
['?','?','?','?','?','?'], ['?','?','?','?','?','?'], ['?','?','?','?','?','?']]
['?','?','?','?','?','?'), ['?','?','?','?','?','?']]

steps of candidate elimination algorithm 3
[['sunny','warm','?','strong','?','?']
[['sunny','?','?','?','?','?'], ['?','warm','?','?','?','?'],
['?','?','?','?','?','?'], ['?','?','?','?','?','?']]

Date.......................

Expt. No.................... Page No.......................

steps of candidate elimination algorithm 4

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

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

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

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

final specific hypothesis

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

final general hypothesis

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

3. write a program to demonstrate the working of the decision tree based ID3 algorithm. use an appropriate data set for binding the decision tree and apply this knowledge to classify a new sample

```
import pandas as pd.
from pandas import Dataframe.
df_tennis = pd.read-csv('c:\\user\\Admin\\Desktop\\
                Albin\\playtennis.csv')
attribute-names = list(df_tennis.colums)
attribute_names.remove('play Tennis').
print(attribute_names).
def entropy-of-list(lst):
        from collections import counter.
        count= counter(x for x in lst)
        num_instances = len(lst)*1.
        probs = [x/num_instances for x in count.values()]
        return entropy(probs) .
def entropy (probs):
    import math
        return sum([-prob*math.log(prob,2) for
            prob in probs]).
Total_entropy =entropy-of-list (df_tennis['
 playTennis'])
def information_gain (df, split_attribute_name,
 target_attribute_name, trace=0):
    df_split= df.groupby (split_attribute_name)
    nobs = len(df.index)*1.
```

```
df_agg_ent = df_split_agg({target_attribute_name:
        [entropy_of_list, lamda x: len(x)/nobs]})
df_agg_ent.columns = ['Entropy','prop observations']
new_entropy = sum(df_agg_ent['entropy'] * df_
                  agg_ent['propobservations']).
old_entropy = entropy_of_list (df[target_
                  attribute_name])
print(split_attribute_name,'Ig :', old_entropy_new
        entropy).
  return old_entropy new_entropy.
def id3(df, target_attribute_name, attribute_
  names, default_class = None):
   from collections import counter:
   count = counter( x for x in df [target_attribute_
    name])
  if len (count)==1:
    return next(iter count))
  elif df_empty or (not attribute_names):
      return default_class.
  else:
     default_class = max(count.key()).
     gain = [information_gain(df, attr, target_
     attribute_name) for atter in attribute_
     names]
     print(")
     index_of_max = gain.index(max(gain))
     best_attr = attribute.names [index_of_max]
     tree = {best_attr:{}}
  remaining_attribute_names =[i for i in attribute_
```

```
name if 1} = best_attr}
for attr_val , data_subset in df.groupby.
(best attr):
    subtree = ID3(data_subset, target_attribute_
    name, remaining_attribute_names_default_
    class).
   tree [best attr][attr_val] = subtree

return.

from pprint import pprint.
tree = ID3(df_tennis, 'playtennis', attribute_
names).
print("\n\n The resultant decision tree is:
\n") pprint(tree)
```

Date.........

Page No.........

Expt. No..................

output

['outlook', 'temperature', 'humidity', 'wind'].

outlook IG = 0.2467498197744391

Temperature IG : 0.029222565658954647.

Humidity IG : 0.15183550136234136.

wind IG : 0.04812703040826927.

Temperature IG : 0.01997309402197489.

Humidity IG : 0.01997309402197489.

wind IG : 0.9709505944546686.

Temperature IG : 0.5709505944546686.

Humidity IG : 0.9709505944546686

wind IG : 0.01997309402197489.

The result of decision tree is

{'outlook': {'overcast': 'yes',
'Rain': {'wind': {'strong': 'No', 'weak': 'yes'}},
'sunny': {'Humidity': {'High': 'No', 'Normal':
'yes'}}}}