Experiment-2

AIM:

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

Description:

1. Initialize h to the most Specific hypothesis in H that fits all the positive examples.
2. For each positive training instance x
   For each attribute constraint ai in h
   If the constraint ai is satisfied by x
   Then do nothing
   Else replace ai in h by the next more general Constraint that is satisfied by x
3. Take the next example and if it is negative, then no change occur to the hypothesis.
4. Keep repeating the above steps till all the training examples are complete
5. After we have completed all the training examples we will have the final hypothesis
6. Output hypothesis h

Training Examples:

Example	isSky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Same	No
4	Sunny	Warm	High	Strong	Warm	Change	Yes

Source Code:

```
import csv
num_attributes=6
a=[]
print("\n the given Training data set \n");
with open('enjoysport.csv','r') as csvfile:
    reader=csv.reader(csvfile)
    for row in reader:
        a.append(row)
        print(row)
print("\n The initial value of hypothesis:")
hypothesis=['0']*num_attributes
print(hypothesis)
for j in range(0,num_attributes):
    hypothesis[j]=a[0][j];
printf("\n Find S: Finding a Maximally specific
    Hypothesis \n")
for i in range(0,len(a)):
    if a[i][num_attributes]=='yes':
```

```
for j in range(0, num_attributes):
    if a[i][j] != hypothesis[j]:
        hypothesis[j] = '?'
    else:
        hypothesis[j] = a[i][j]
print("For Training instance No:{0} the hypothesis is :", format(i), hypothesis)
print("\n The Maximally specific Hypothesis for a given Training Examples :\n")
print(hypothesis)
```

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

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 initial value of hypothesis :

['0', '0', '0', '0', '0', '0']

Find S: Finding a Maximally specific Hypothesis

For Training Example NO:0 the hypothesis is
['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

For Training Example NO:1 the hypothesis is
['sunny', 'warm', '?', 'strong', 'warm', 'same']

For Training Example NO:2 the hypothesis is
['sunny', 'warm', '?', 'strong', 'warm', 'same']

For Training Example NO:3 the hypothesis is
['sunny', 'warm', '?', 'strong', '?', '?']

The Maximally specific hypothesis for given Training Examples:

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

Experiment-3

AIM:

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.

Description:

Initialize G to the set of Maximally

Initialize S to the set of Maximally

For each instance training example d, do

- If d is a positive example
- Remove from G any hypothesis inconsistent with d
- For each hypothesis s in S that is not consistent with d
- Remove s from S
- Add to S all minimal generalizations h of S such that h is consistent with d, and some member of G is more general than h
- Remove from S any hypothesis that is more general than another hypothesis in S
- If d is a negative example
- Remove from S any hypothesis inconsistent with d for each hypothesis g in G that is not consistent with d
- Remove g from G
- Remove from G any hypothesis that is less general than another hypothesis in G.

Initializing the version space to set all hypotheses in H

When first training example is presented, the candidate-elimination algorithm checks S-boundary.

Finds that is overly specif and it fails to cover the positive example.

The boundary is revised by moving it to the least more general hypothesis that covers this new example.

No update of the G boundary is need.

When second training example is observed. It has similar effect of generalizing S further S further to S2, leaving G again unchanged i.e $G_2 = G_1 = G_0$

steps repeated, till fourth example

After processing four examples, the boundary sets S_4 and G_4 delimit the version space of all hypotheses Consistent with set of incrementally observed.

Source Code

```
import numpy as np
import pandas as pd
data = pd.read_csv('2.csv')
Concepts = np.array(data.iloc[:,0:-1])
```

```
target =np.array(data.iloc[:,-1])
def learn(concepts, target):
    Specific_h = concepts[0].copy()
print("intilialization of specific_h \n", specific_h)
general_h =[["?" for i in range(len(specific_h))]
for i in range(len(specific_h))]
print("initialization of general_h \n", general_h)
for i, h in enumerate(concepts):
    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("step {}".format(i+1))
print(specific_h)
print(general_h)
print("\n")
print("\n")
indices =[i for i, val in enumerate(general_h)
  if val == ['?','?','?','?','?','?']]
  retun 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")
```

Output

```
Initialization of specific_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
Initialization of general_h
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
If Instance is positive
step 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?']
If Instance is positive
Step 2.
['sunny' 'warm' ? 'strong' 'warm' 'same']
['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?'] ['?', '?', '?', '?', '?', '?']
If Instance is Negative
Step 3
```

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

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

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

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

If instance is positive

Step 4

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

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

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

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

Final specific h:

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

Final General h:

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

AIM:

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.

Description:

Examples are the training examples.
Target_attribute is the attribute whose value is to be predicted or by the tree.
Attributes is a list of other attributes that may be tested by learned decision tree.
Returns a decision tree that correctly classifier the given Examples.

Create a Root node for the tree.
If all Examples are positive, Return the single node tree Root, With label = +
If all examples are Negative, Return the single node tree Root With label = -
Add a new root to below branch root.
= most common value of Target_attributes in Examples.
end and return root

Entropy:

Entropy measures the impurity of all collection of examples:

$$\text{Entropy}(S) \equiv -P_{\oplus}\log_2 P_{\oplus} - P_{\ominus}\log_2 P_{\ominus}$$

→ P_+ is proportion of positive examples in S
→ P_- is proportion of negative example in S

Information Gain:

is expected reduction in entropy caused by partitioning the examples according to this Attribute

The information gain, Gain (S, A)

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Training Dataset:

Day	Outlook	Temperature	Humidity	Wind	play Tennis
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_3	Overcast	Hot	High	Weak	Yes
D_4	Rain	Mild	High	Weak	Yes
D_5	Rain	Cool	Normal	Weak	Yes
D_6	Rain	Cool	Normal	Strong	No
D_7	Overcast	Cool	Normal	strong	Yes
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{10}	Rain	Mild	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes
D_{12}	Overcast	Mild	High	strong	Yes
D_{13}	Overcast	Hot	Normal	Weak	Yes
D_{14}	Rain	Mild	High	Strong	No

Day	Outlook	Temperature	Humidity	Wind
T_1	Rain	Cool	Normal	Strong
T_2	Sunny	Mild	Normal	Strong