Practical No: 4

Candidate-elimination algorithm

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:

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

- You can consider this as an extended form of Find-S algorithm.
- Consider both positive and negative examples.
- Actually, positive examples are used here as Find-S algorithm (Basically they are generalizing from the specification).
- While the negative example is specified from generalize form.

Terms :-

General Hypothesis: Not Specifying features to learn the machine.

G = {'?', '?','?','?'…}: Number of attributes.

Specific Hypothesis: Specifying features to learn machine (Specific feature).

S= {'pi','pi','pi'…}: Number of pi depends on number of attributes.

Version Space: It is intermediate of general hypothesis and Specific hypothesis. It not only just written one hypothesis but a set of all possible hypothesis based on training data-set.

Candidate-elimination algorithm :-

Step1: Load Data set

Step2: Initialize General Hypothesis and Specific Hypothesis.

Step3: For each training example

Step4: If example is positive example

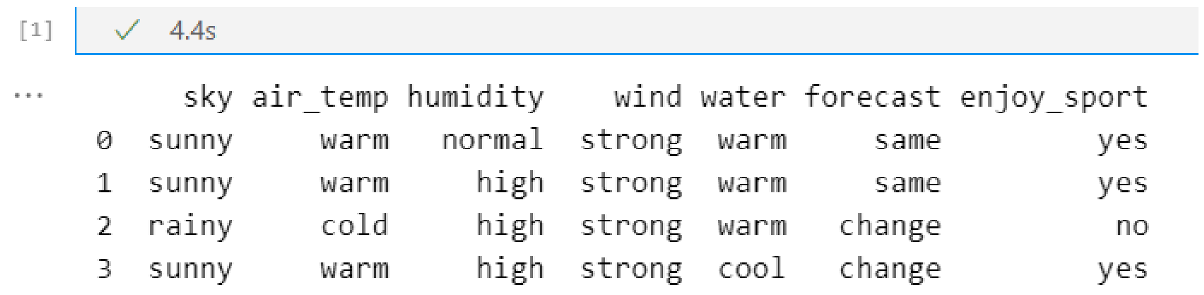
```
    if attribute_value == hypothesis_value:
        Do nothing
      else:
        replace attribute value with '?' (Basically generalizing it)
```

Step5: If example is Negative example

Make generalize hypothesis more specific.

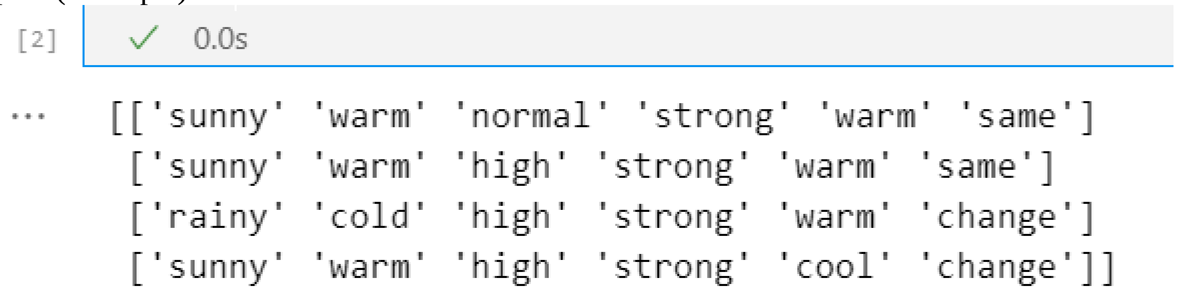
Code and output :-

```
import numpy as np
import pandas as pd

#Loading data from a csv file.
data = pd.DataFrame(data=pd.read_csv('enjoysport.csv'))
print(data)
```



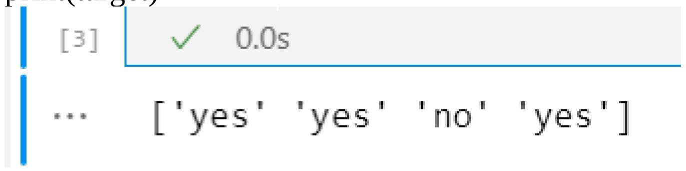
```
[1]  ✓ 4.4s

...       sky air_temp humidity    wind water forecast enjoy_sport
     0  sunny     warm   normal  strong  warm     same         yes
     1  sunny     warm     high  strong  warm     same         yes
     2  rainy     cold     high  strong  warm   change          no
     3  sunny     warm     high  strong  cool   change         yes
```

```
#Separating concept features from Target
concepts = np.array(data.iloc[:,0:6])
print(concepts)
```



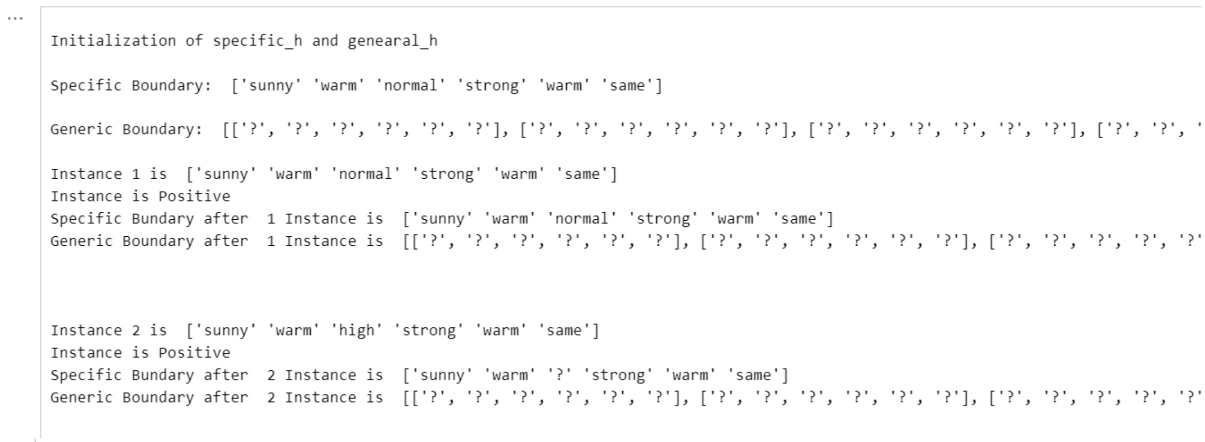
```
[2]  ✓ 0.0s

...  [['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
      ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
      ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
      ['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
```

```
#Isolating target into a separate DataFrame
#Copying last column to target  array
target = np.array(data.iloc[:,6])
print(target)
```

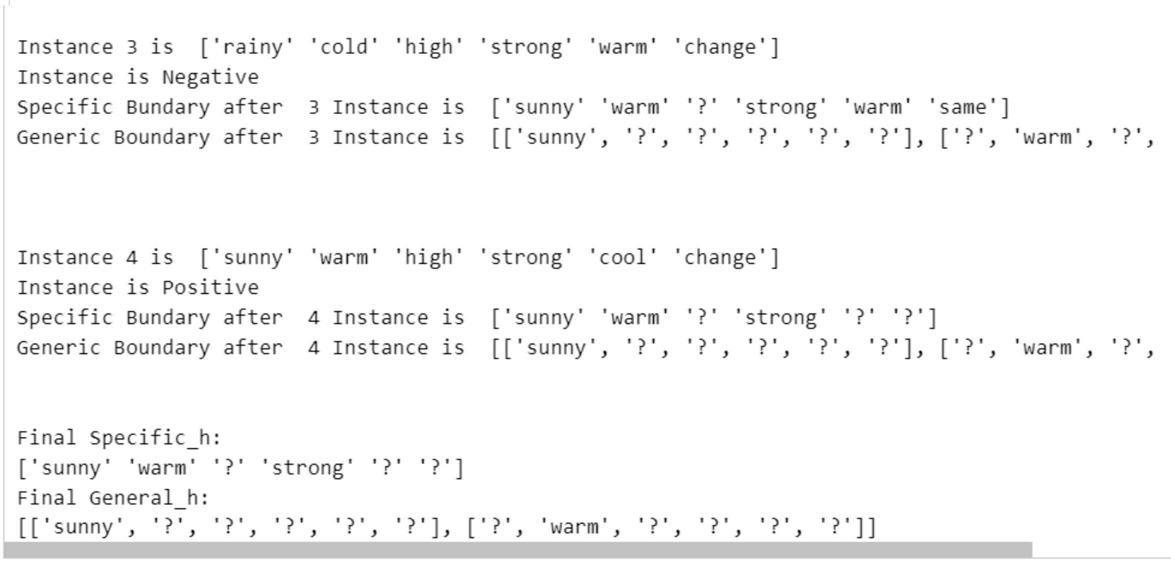


```
[3]  ✓ 0.0s

...  ['yes' 'yes' 'no' 'yes']
```

```
def learn(concepts, target):
#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()
   print("\nInitialization of specific_h and genearal_h")
   print("\nSpecific Boundary: ", specific_h)
   general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
   print("\nGeneric Boundary: ",general_h)
# The learning iterations.
   for i, h in enumerate(concepts):
      print("\nInstance", i+1 , "is ", h)
# Checking if the hypothesis has a positive target.
      if target[i] == "yes":
         print("Instance is Positive ")
         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":
         print("Instance is Negative ")
         for x in range(len(specific_h)):
# For negative hypothesis change values only in G.
            if h[x]!= specific_h[x]:
               general_h[x][x] = specific_h[x]
            else:
               general_h[x][x] = '?'

      print("Specific Bundary after ", i+1, "Instance is ", specific_h)
      print("Generic Boundary after ", i+1, "Instance is ", general_h)
      print("\n")
# 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 Specific_h: ", s_final, sep="\n")
print("Final General_h: ", g_final, sep="\n")
```



```
Initialization of specific_h and genearal_h

Specific Boundary:  ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Generic Boundary:  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '

Instance 1 is  ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
Instance is Positive
Specific Bundary after  1 Instance is  ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
Generic Boundary after  1 Instance is  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'



Instance 2 is  ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
Instance is Positive
Specific Bundary after  2 Instance is  ['sunny' 'warm' '?' 'strong' 'warm' 'same']
Generic Boundary after  2 Instance is  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'
```



```
Instance 3 is  ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
Instance is Negative
Specific Bundary after  3 Instance is  ['sunny' 'warm' '?' 'strong' 'warm' 'same']
Generic Boundary after  3 Instance is  [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?',



Instance 4 is  ['sunny' 'warm' 'high' 'strong' 'cool' 'change']
Instance is Positive
Specific Bundary after  4 Instance is  ['sunny' 'warm' '?' 'strong' '?' '?']
Generic Boundary after  4 Instance is  [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?',


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