6/2/23

EXPERIMENT - 2 A

AIM

To implement the Find-S algorithm in Python

ALGORITHM

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute constraint a, in h
   If the constraint a, is satisfied by x
   Then do nothing
   Else replace a, in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

THEORY

The Find-S algorithm is a basic concept learning algorithm in ML. It is used to find the maximally specific hypothesis. Find-S starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive example.

OUTPUT

```
PS C:\Users\DELL\Downloads> & "C:/Program Files/Python311/python.exe"

 Data read from dataset.csv :

      Time Weather Temperature Company Humidity    Wind Goes
0  Morning   Sunny        Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning   Sunny    Moderate     Yes   Normal  Normal  Yes
3  Evening   Sunny        Cold     Yes     High  Strong  Yes

 Storing attributes in numpy array :

 [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

 Target =  ['Yes', 'No', 'Yes', 'Yes']

 Final Hypothesis =  ['?' 'Sunny' '?' 'Yes' '?' '?']
```

CODE

```
import pandas as pd
import numpy as np

data = pd.read_csv ( "ML Lab Files/dataset.csv" )
print ( "\n Data read from dataset.csv :\n\n" , data )

arr = np.array ( data )
attributes = arr [ : , : -1 ]
print ( "\n Storing attributes in numpy array :\n\n" , attributes )

target = [ x [ -1 ] for x in arr ]
print ( "\n Target = " , target )

for ind , val in enumerate ( target ) :
    if val == "Yes" :
        specific_hypothesis = attributes [ ind ].copy ()
        break

for ind , val in enumerate ( attributes ) :
    if target [ ind ] == "Yes" :
        for el in range ( len ( specific_hypothesis ) ) :
            if val [ el ] != specific_hypothesis [ el ] :
                specific_hypothesis [ el ] = '?'

print ( "\n Final Hypothesis = " , specific_hypothesis )
```

RESULT

Hence, the Find-S algorithm has been implemented successfully

6/2/23

EXPERIMENT - 2B

AIM

To implement the Naïve Bayes' classifier algorithm in Python

ALGORITHM

* Convert the given dataset into frequency tables
* Generate likelihood table by finding the probabilities of given features
* Now, use Bayes' theorem to calculate the posterior probability

THEORY

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes' theorem and used for solving classification problems. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Bayes' theorem is used to determine the probability of a hypothesis depending on conditional probability. It is given as,

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

OUTPUT

```
1       Rainy      Hot     High   True     No
2    Overcast      Hot     High  False    Yes
3       Sunny     Mild     High  False    Yes
4       Sunny     Cool   Normal  False    Yes
5       Sunny     Cool   Normal   True     No
6    Overcast     Cool   Normal   True    Yes
7       Rainy     Mild     High  False     No
8       Rainy     Cool   Normal  False    Yes
9       Sunny     Mild   Normal  False    Yes
10      Rainy     Mild   Normal   True    Yes
11   Overcast     Mild     High   True    Yes
12   Overcast      Hot   Normal  False    Yes
13      Sunny     Mild     High   True     No

 Numpy array =  [['Rainy' 'Hot' 'High' False 'No']
 ['Rainy' 'Hot' 'High' True 'No']
 ['Overcast' 'Hot' 'High' False 'Yes']
 ['Sunny' 'Mild' 'High' False 'Yes']
 ['Sunny' 'Cool' 'Normal' False 'Yes']
 ['Sunny' 'Cool' 'Normal' True 'No']
 ['Overcast' 'Cool' 'Normal' True 'Yes']
 ['Rainy' 'Mild' 'High' False 'No']
 ['Rainy' 'Cool' 'Normal' False 'Yes']
 ['Sunny' 'Mild' 'Normal' False 'Yes']
 ['Rainy' 'Mild' 'Normal' True 'Yes']
 ['Overcast' 'Mild' 'High' True 'Yes']
 ['Overcast' 'Hot' 'Normal' False 'Yes']
 ['Sunny' 'Mild' 'High' True 'No']]

 [0.028218694885361547, 0.0]
1.0  >  0.0  => Result : Positive Sample !
```

CODE

```
import pandas as pd
import numpy as np

data = pd.read_csv ( "ML Lab Files/nb_dataset.csv" )
print ( "\n Data read from dataset.csv :\n\n" , data )

arr = np.array ( data )
print ( "\n Numpy array = " , arr )
X = arr [ : , : -1 ]
y = arr [ : , -1 ]

iters = [ arr [ : , x ] for x in range ( 4 ) ]
attr_labels = [ list ( set ( itr ) ) for itr in iters ]

def computeTable ( iters , attr_labels ) :
    matrix = []
    for itr , attr_label in zip ( iters , attr_labels ) :
        mtx = { x : [ 0 for y in range ( 2 ) ] for x in attr_label }
        for el in range ( len ( itr ) ) :
            for lb in attr_label :
                if itr [ el ] == lb :
                    if y [ el ] == "Yes" :
                        mtx [ lb ] [ 0 ] += 1
                    else :
                        mtx [ lb ] [ 1 ] += 1
        matrix.append ( mtx )
    return matrix

mtcs = computeTable ( iters , attr_labels )

targets = [ 0 , 0 ]
for el in y :
    if el == "Yes" :
        targets [ 0 ] += 1
    else :
        targets [ 1 ] += 1
targets = [ target / sum ( targets ) for target in targets ]

today = [ "Overcast" , "Hot" , "Normal" , False ]

def calcProbability ( matrix , target ) :
    probs = [ 1 , 1 ]
    for ind in range ( len ( matrix ) ) :
        countYes = matrix [ ind ] [ today [ ind ] ] [ 0 ]
        countNo = matrix [ ind ] [ today [ ind ] ] [ 1 ]
        totalYes = sum ( [ matrix [ ind ][ x ] [ 0 ] for x in matrix [ ind ] ] )
        totalNo = sum ( [ matrix [ ind ] [ x ] [ 1 ] for x in matrix [ ind ] ] )
        currProbs = [ countYes / totalYes , countNo / totalNo ]
        probs = [ prob * currProb for prob , currProb in zip ( probs , currProbs ) ]
    probs = [ prob * trg for prob , trg in zip ( probs , target ) ]
    return probs
```

```
result = calcProbability ( mtcs , targets )
print ( "\n", result )
resYes , resNo = result [ 0 ] / sum ( result ) , result [ 1 ] / sum ( result )
if resYes == resNo :
    print ( resYes , " = " , resNo , " => Result : Positive or Negative Sample !" )
elif resYes > resNo :
    print ( resYes , " > " , resNo , " => Result : Positive Sample !" )
else :
    print ( resYes , " < " , resNo , " => Result : Negative Sample !" )
```

RESULT

Hence, the Naive Bayes algorithm has been implemented successfully