Output :

```
data
   sky    airtemp  humidity  wind    water  forcast  enjoysport
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

concepts
 array([['sunny', 'warm', 'normal', 'strong', 'warm', 'same'],
        ['sunny', 'warm', 'high', 'strong', 'warm', 'same'],
        ['rainy', 'cold', 'high', 'strong', 'warm', 'change'],
        ['sunny', 'warm', 'high', 'strong', 'cool', 'change']],
      dtype = object)

target
 array(['yes', 'yes', 'no', 'yes'], dtype = object)

print (train (concepts , target))
 ['sunny' 'warm' '?' 'strong' '?' '?']
```

1 Implement and demostrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the data from a .csv file (enjoySport Dataset)

```
import pandas as pd
import numpy as np
data = pd.read_csv (r" c:\users\enjoysport.csv")
data
concepts = np.array(data)[:,:-1]
concepts
target = np.array(data)[:,-1]
target
def train(con, tar):
    for i, val in enumerate(tar):
        if val == 'yes' :
            specific_h = con[i].copy()
            break
    for i, val in enumerate(con):
        if tar[i] == 'yes' :
            for x in range(len(specific_h)):
                if val[x] != specific_h[x]:
                    specific_h[x] = '?'
                else
                    pass
    return specific_h

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

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

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

Experiment No..................

2 For a given set of training data examples stored in a .CSV file (enjoysport Dataset), implement & demonstrate the candidate - Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
import pandas as pd
import numpy as np

data = pd.read_csv(r" c:\users\enjoysport.csv")
concepts = np.array (data. iloc [:, 0:-1])
target = np.array(data.iloc [:,-1])

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization 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] = '?'
```

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

```
        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 ==
    ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    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")
```

```
    Outlook    Temperature  Humidity  Windy  Play-Tennis
0   Sunny      Hot          High      False  No
1   Sunny      Hot          High      True   No
2   overcast   Hot          High      False  Yes
3   Rainy      mild         High      False  Yes
4   Rainy      Cool         Normal    False  Yes
5   Rainy      cool         Normal    True   No
6   overcast   Cool         Normal    True   Yes
7   Sunny      mild         High      False  NO
8   Sunny      cool         Normal    False  Yes
9   Rainy      mild         Normal    False  Yes
10  Sunny      mild         Normal    True   Yes
11  overcast   mild         High      True   Yes
12  overcast   Hot          Normal    False  Yes
13  Rainy      mild         High      True   NO
```

```
14
```

```
(14, 5)
```

```
0
   outlook    Temperature  Humidity  Windy  Play-Tennis
0  Sunny      Hot          High      False  No
1  Sunny      Hot          High      True   NO
2  overcast   Hot          High      False  Yes
3  Rainy      mild         High      false  Yes
4  Rainy      cool         Normal    False  Yes
```

```
    outlook    Temperature  Humidity  Windy  Play-tennis
9   Rainy      mild         Normal    False  Yes
10  Sunny      mild         Normal    True   Yes
11  overcast   mild         High      True   Yes
12  overcast   Hot          Normal    False  Yes
13  Rain       mild         High      True   NO
```

3 Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree & apply the Knowledge to classify a new sample (Play Tennis Dataset)

3:
```
Import numpy as np
import pandas as pd
from sklearn import metrics

df = pd.read_csv(r" c:\users\PlayTennis.csv")
Value = ['outlook', 'Temperature', 'Humidity', 'windy']
df

len(df)
df.shape
df.head()
df.tail()
df.describe()

from sklearn Import preprocessing
string_to_int = preprocessing.LabelEncoder()
df = df.apply(string_to_int.fit_transform)
df

feature_cols = ['outlook', 'Temperature', 'humidity', 'windy']
x = df[feature_cols]
y = df.Play-Tennis
```

```
        outlook   Temperature   Humidity   windy   Play-Tennis
count     14          14           14        14        14
unique    3           3            2         2         2
top      Sunny       mild         High     False      Yes
freq      5           6            7         8         9
```

```
     outlook   Temperature   Humidity   Windy   Play-tennis
0       2           1            0         0         0
1       2           1            0         1         0
2       0           1            0         0         1
3       1           2            0         0         1
4       1           0            1         0         1
5       1           0            1         1         0
6       0           0            1         1         1
7       2           2            0         0         0
8       2           0            1         0         1
9       1           2            1         0         1
10      2           2            1         1         1
11      0           2            0         1         1
12      0           1            1         0         1
13      1           2            0         1         0
```

Accuracy : 0.8

```
     Actual   Predicted
13     0          1
0      0          0
4      1          1
9      1          1
2      1          1
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size
                                                    = 0.30)

from sklearn.tree import DecisionTreeclassifier
classifier = DecisionTreeclassifier(criterion = "entropy", random_state=1
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

data_p = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

from sklearn.metrics import classification_report, confusion_matri
print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

```
[[1 1]
 [0 3]]

             precision    recall    f1-score    support

    0          1.00        0.50       0.67         2
    1          0.75        1.00       0.86         3

avg / total    0.85        0.80       0.78         5
```

	Outlook	Temperature	Humidity	Windy	Play-Tennis
0	Sunny	Hot	High	False	NO
1	Sunny	Hot	High	True	NO
2	Overcast	Hot	High	False	yes
3	Rainy	mild	High	False	yes
4	Rainy	cool	Normal	False	yes
5	Rainy	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	yes
7	Sunny	mild	High	False	NO
8	Sunny	cool	Normal	False	Yes
9	Rainy	mild	Normal	false	yes
10	Sunny	Mild	Normal	True	yes
11	Overcast	mild	High	True	yes
12	Overcast	Hot	Normal	false	Yes
13	Rainy	mild	High	True	NO

14

(14, 5)

	Outlook	Temperature	Humidity	Windy	Play-Tennis
0	Sunny	Hot	High	False	NO
1	Sunny	Hot	High	True	NO
2	Overcast	Hot	High	False	Yes
3	Rainy	mild	High	False	yes
4	Rainy	cool	Normal	False	yes

	Outlook	Temperature	Humidity	Windy	Play-Tennis
9	Rainy	mild	Normal	False	yes
10	Sunny	Mild	Normal	True	yes
11	Overcast	mild	High	True	Yes
12	Overcast	Hot	Normal	False	yes
13	Rainy	mild	High	True	NO

4 Write a program to demonstrate the working of the decision Tree based CART algorithm. (Play Tennis Dataset)

```
import numpy as np
import pandas as pd
from sklearn import metrics

df= pd.read_csv(r" c:\users\PlayTennis.csv")
value = ['Outlook', 'Temperature', 'Humidity', 'windy']
df

len(df)
df.shape
df.head()
df.tail()
df.describe()

from sklearn import preprocessing
string_to_int = preprocessing.LabelEncoder()
df = df.apply(string_to_int.fit_transform)
df

feature_cols = ['Outlook', 'Temperature', 'Humidity', 'windy']
x = df[feature_cols]
y = df.play_Tennis

from sklearn.model_selection import train_test_split
x_Train, x_test, y-train, y-Test = train_test_split(x,y,test_size=0.30
       outlook  Temperature  Humidity  windy  Play-Tennis
count    14        14           14       14       14
unique    3         3            2        2        2
top     Sunny     Mild         High     False     Yes
freq      5         6            7        8        9
```

```
    outlook  Temperature  Humidity  windy  Play-Tennis
0      2          1           0        0        0
1      2          1           0        1        0
2      0          1           0        0        1
3      1          2           0        0        1
4      1          0           1        0        1
5      1          0           1        1        0
6      0          0           1        1        1
7      2          2           0        0        0
8      2          0           1        0        1
9      1          2           1        0        1
10     2          2           1        1        1
11     0          2           0        1        1
12     0          1           1        0        1
13     1          2           0.       1        0
```

```
Accuracy : 1.0

     Actual  Predicted
10     1        1
5      0        0
9      1        1
6      1        1
12     1        1
```

```
from sklearn.tree import DecisionTreeclassifier
classifier = DecisionTreeclassifier(criterion = 'gini', random_state = 100
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

data_p = pd.DataFrame({'Actual': y_test, 'Predicted':y_pred
data_p

from sklearn.metrics import classification_report, confusion_matri
print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

```
[[1 0
  0 4]]

             precision   recall   f1-score   support
   0           1.00       1.00      1.00        1
   1           1.00       1.00      1.00        4
avg/total      1.00       1.00      1.00        5
```

Experiment No.................

```
14

(14, 5)

     Outlook   Temperature  Humidity   Windy
0    Sunny     Hot          High       False
1    Sunny     Hot          High       True
2    overcast  Hot          High       False
3    Rainy     Mild         High       False
4    Rainy     cool         Normal     False
5    Rainy     cool         Normal     True
6    overcast  cool         Normal     True
7    Sunny     Mild         High       False
8    Sunny     cool         Normal     False
9    Rainy     Mild         Normal     False
10   Sunny     Mild         Normal     True
11   overcast  Mild         High       True
12   overcast  Hot          Normal     False
13   Rainy     Mild         High       True

0    25
1    30
2    46
3    45
4    52
5    23
6    43
7    35
8    38
9    46
10   48
11   52
12   44
13   30
Name: Golf Players, dtype: int64
```

5 Write a program to demonstrate Decision tree regression for a given dataset.

```
import numpy as np
import pandas as pd
from sklearn import metrics

df = pd.read_csv(r"c:\users\tennisdataReg.csv")

len(df)

df.shape

x = df.drop("Golf Players", axis = 1)
y = df['Golf players']

x

y

from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
string_to_int = preprocessing.LabelEncoder()
x = x.apply(string_to_int.fit_transform)
x

from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor()
reg = reg.fit(x, y)
```

	Outlook	Temperature	Humidity	windy
0	2	1	0	0
1	2	1	0	1
2	0	1	0	0
3	1	2	0	0
4	1	0	1	0
5	1	0	1	1
6	0	0	1	1
7	2	2	0	0
8	2	0	1	0
9	1	2	1	0
10	2	2	1	1
11	0	2	0	1
12	0	1	1	0
13	1	2	0	1

```
Result is : [30.]

Result is : [25.]
```

```
y-pred = reg.predict([[2,1,0,1]])
print ("Result is :", y-pred)

y-pred = reg.predict([[2,1,0,0]])
print ("Result is :", y-pred)
```

```
Training completed in 1 epochs
Final weights : w1 = 1.2, w2 = 1.1, bias = -1.0
Testing perceptron for AND gate :
Input : [0 0] , output : 0, Expected : 0
Input : [0 1], output : 0, Expected : 0.
Input : [1 0], output : 0, Expected : 0
Input : [1 1], output : 1. Expected : 1
```

6 Implement a Perceptron algorithm for AND logic gate with 2-bit binary Input. Test for different hyper parameters

```
import numpy as np

inputs = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])

w1, w2 = 1.2, 0.6
bias = -1.0
threshold = 1
learning_rate = 0.5

def activation_function(net_input):
    return 1 if net_input >= threshold else 0

epochs = 0
while True:
    error count = 0
    for i in range(len(inputs)):
        net_input = w1 + inputs[i][0] + we *inputs[i][1] + bias
        output = activation_function(net_input)
        error = expected_output[i] - output
        if error != 0:
            w1 += learning_rate * error * inputs [i][0]
            we += learning_rate * error * inputs [i][1]
            bias + = learning rate * error
            error count + = 1

        epochs + =1

        if error count = = 0 :
            break

print( f" Training completed in {epochs} epochs")
print (f" Final weights : w1 ={w1}, w2= {w2}, bias = {bias}")

print (" Testing perception for AND gate : ")
for i in range(len(inputs)) :
    net_input = w1 * inputs [i][0] + w2 * inputs [i][1] + bias
    output = activation_function (net_input)
    print (f" Input : {inputs [i]}, Output : {output},
            Expected : {expected_outputs [i] }")
```