Program 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 and apply this knowledge to classify a new sample.

Following terminologies are used in this algorithm

- **Entropy :** Entropy is a measure of impurity

It is defined for a binary class with values a/b as:

$$Entropy = - p(a)*\log(p(a)) - p(b)*\log(p(b))$$

- **Information Gain :** measuring the expected reduction in Entropy

$$Gain(S,A) = Entropy(S) - \sum_{v=1}^{n} (|Sv|/|S|) * Entropy(Sv)$$

THE PROCEDURE

1) In the ID3 algorithm, begin with the original set of attributes as the root node.

2) On each iteration of the algorithm, iterate through every unused attribute of the remaining set and calculates the entropy (or information gain) of that attribute.

3) Then, select the attribute which has the smallest entropy (or largest information gain) value.

4) The set of remaining attributes is then split by the selected attribute to produce subsets of the data.

5) The algorithm continues to recurs on each subset, considering only attributes never selected before.

Dataset Details

playtennis dataset which has following structure
Total number of instances=15
Attributes=Outlook, Temperature, Humidity, Wind, Answer
Target Concept=Answer

ID3 (Learning Sets S, Attributes Sets A, Attributes values V) Return Decision Tree

Begin
Load learning sets S first, create decision tree root node 'rootNode', add learning set S into root node as its subset
For rootNode,
1) Calculate entropy of every attribute using the dataset

2) Split the set into subsets using the attribute for which entropy is minimum (or information gain is maximum)

3) Make a decision tree node containing that attribute

4) Recurse on subsets using renaming attributes

End

This approach employs a top-down, greedy search through the space of possible decision trees.

- Algorithm starts by creating root node for the tree
- If all the examples are positive then return node with positive label
- If all the examples are negative then return node with negative label
- I f Attributes is empty, Return the single-node tree Root, with label = most common value of Targetattribute in Example
- Otherwise -

1. Calculate the entropy of every attribute using the data set S using formula

Entropy = - p(a)*log(p(a)) - p(b)*log(p(b))

2. Split the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum) using formula

Gain(S,A)= Entropy(S) - Sum for v from 1 to n of (|Sv|/|S|) * Entropy(Sv)

3. Make a decision tree node containing that attribute

4. Recurring on subsets using remaining attributes.

```
import ast
import csv
#import sys
import math
import os
def load_csv_to_header_data(filename):
    path = os.path.normpath(os.getcwd() + filename)
    ''' os.path.normpath(path)
Normalize a pathname by collapsing redundant separators and up-level references so that A//B,
A/B/, A/./B and A/foo/../B all become A/B. This string manipulation may change the meaning
of a path that contains symbolic links. On Windows, it converts forward slashes to backward
slashes. To normalize case, use normcase().'''
    print(path)
    fs = csv.reader(open(path))
    all_row = []
    for r in fs:
        all_row.append(r)
    headers = all_row[0]
    idx_to_name, name_to_idx = get_header_name_to_idx_maps(headers)
    data = { 'header': headers,'rows': all_row[1:],'name_to_idx': name_to_idx,'idx_to_name':
idx_to_name}
    return data
def get_header_name_to_idx_maps(headers):
    name_to_idx = {}
    idx_to_name = {}
```

```
    for i in range(0, len(headers)):
        name_to_idx[headers[i]] = i
        idx_to_name[i] = headers[i]
        #print(name_to_idx)
        #print(idx_to_name)
    return idx_to_name, name_to_idx
def project_columns(data, columns_to_project):
    data_h = list(data['header'])
    data_r = list(data['rows'])
    all_cols = list(range(0,len(data_h)))
    columns_to_project_ix = [data['name_to_idx'][name] for name in columns_to_project]
    #print(columns_to_project_ix)
    columns_to_remove = [cidx for cidx in all_cols if cidx not in columns_to_project_ix]
    #print(columns_to_remove)
    for delc in sorted(columns_to_remove, reverse=True):
        del data_h[delc]
        for r in data_r:
            del r[delc]
    idx_to_name, name_to_idx = get_header_name_to_idx_maps(data_h)
    return {'header': data_h, 'rows': data_r,'name_to_idx': name_to_idx,'idx_to_name':
idx_to_name}
def get_uniq_values(data):
    idx_to_name = data['idx_to_name']
    idxs = idx_to_name.keys()
    #print(idxs)
    val_map = {}
    for idx in iter(idxs):
        val_map[idx_to_name[idx]] = set()
    #print(val_map)
    for data_row in data['rows']:
        for idx in idx_to_name.keys():
            att_name = idx_to_name[idx]
            val = data_row[idx]
            if val not in val_map.values():
                val_map[att_name].add(val)
    #print(val_map)
    return val_map
```

INPUTS AND OUTPUTS

Input- Input to the decision algorithm is a dataset stored in .csv file which consists of attributes, examples, target concept.

Output- For the given dataset decision tree algorithm produces the decision tree starting with rootnode which has highest information gain.

```
{ 'data_file' : '//tennis.csv', 'data_mappers' : [], 'data_project_columns' :
['Outlook', 'Temperature', 'Humidity', 'Windy', 'PlayTennis'], 'target_attribute' :
'PlayTennis'}
```

```
{'data_file': '//tennis.csv', 'data_mappers': [], 'data_project_columns':
['Outlook',
'Temperature', 'Humidity', 'Windy', 'PlayTennis'], 'target_attribute': 'PlayTennis'}
E:\suthan\codes\Machine Learning Lab\3\tennis.csv
{'Outlook', 'Windy', 'Humidity', 'Temperature'}
IF Outlook EQUALS Sunny AND Humidity EQUALS Normal THEN Yes
IF Outlook EQUALS Sunny AND Humidity EQUALS High THEN No
IF Outlook EQUALS Rainy AND Windy EQUALS True THEN No
IF Outlook EQUALS Rainy AND Windy EQUALS False THEN Yes
IF Outlook EQUALS Overcast THEN Yes
```