```
import numpy as np
import pandas as pd
from pprint import pprint
data = pd.read_csv("weather.csv")
data_size= len(data)
treenodes = []
tree = {"ROOT": data}
def total_entropy(data, col):
    mydict = {}
    for elem in data[col]:
        if elem in mydict.keys():
            mydict[elem] += 1
        else:
            mydict[elem] = 1
    total = sum(mydict.values())
    E = 0
    for key in mydict.keys():
        E += entropy(mydict[key], total)
    return E

def entropy(num, denom):
    return -(num/denom) * np.log2(num/denom)

def get_sorted_data(data, column):
    sort = {}
    for column_name in get_attributes(data, column):
        sort[column_name] = data.loc[data[column]==column_name]
    return sort
def get_attributes(data, column):
```

```
    return data[column].unique().tolist()

def InfoGain(total_entropy, sorted_data, entropy_by_attribute):
    length = data_size
    total = 0
    for col, df in sorted_data.items():
        total += (len(df) / length) * entropy_by_attribute[col]
    return total_entropy - total

def get_entropy_by_attribute(sorted_data):
    entropies = {}
    for key, df in sorted_data.items():
        entropies[key] = total_entropy(df, 'Decision')
    return entropies

def drop_node(data, column):
    return data.drop(column, axis=1)

def id3(tree):
    for branch, data in tree.items():

        # Make sure it's a DataFrame
        if not isinstance(data, pd.DataFrame):
            continue

        #Fetch column names so you can use them to iterate later
        columns = data.columns

        # Calculate the Entropy for the entire dataset
```

```
    total_entropy_for_data = total_entropy(data.values, -1)

    # If only one column is left, it means we're done.
    if len(columns) == 1:
        break

    # Keep track of information gain to choose the attribute with maximum info gain.
    info_gain_list = []

    # Now iterate over each column to calculate information gain w.r.t o/p column
for i in range(0, len(data.columns)-1):
            sorted_rows = get_sorted_data(data, columns[i])
            # Calculate the entropy w.r.t to each attribute based on sorted columns
            entropy_by_attribute = get_entropy_by_attribute(sorted_rows)
            # get the info gain
            info_gain = InfoGain(total_entropy_for_data, sorted_rows, entropy_by_attribute)
            # save it
            info_gain_list.append(info_gain)
            # Find index of max info gain
            node = info_gain_list.index(max(info_gain_list))
            # sort the data into branches based on the new node
            branches = get_sorted_data(data, columns[node])
            # If we've reached the end of iterations, just assign the value, else drop the sorted column
for attr, df in branches.items():
        if (total_entropy(df, columns[-1]) == 0):
            branches[attr] = df.iloc[0,-1]
        else:
            branches[attr] = df.drop(columns[node], axis=1)
            # Keep track of nodes already done
```

```
treenodes.append(columns[node])
# add the new branches to the tree
child = {columns[node]: {}}
tree[branch] = child
tree[branch][columns[node]] = branches
   # ID3
id3(tree[branch][columns[node]])
x = id3(tree)
pprint(tree, depth=5)
```