```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
eps = np.finfo(float).eps
from numpy import log2 as log


"""# Decision tree

## Types

1. ID3 (Iterative Dichotomiser 3): It is a simple decision tree algorithm that uses the entropy measure to choose the best attribute for splitting the data.

2. C4.5: It is an extension of the ID3 algorithm that can handle both categorical and numerical data, and can handle missing values in the dataset.

3. CART (Classification and Regression Tree): It is a decision tree algorithm that can be used for both classification and regression tasks. It uses the Gini index as a measure of impurity to choose the best attribute for splitting the data.

#### implement ID3 😁

1. compute the entropy for data-set
2. for every attribute/feature:
       1.calculate entropy for all categorical values
       2.take average information entropy for the current attribute
       3.calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired

# Implement used Functions
"""

def find_entropy(df):
    #target column
    target = df.keys()[-1]
    entropy = 0
    values = df[target].unique()
    #calc entropy
    for value in values:
        fraction = df[target].value_counts()[value]/len(df[target])
        entropy += -fraction*np.log2(fraction)
    return entropy

def average_information(df, attribute):
  target = df.keys()[-1]   #target column
  target_variables = df[target].unique()  #This gives all 'Yes' and 'No'
  variables = df[attribute].unique()    #This gives different features in that attribute (like 'Hot','Cold' in Temperature)
  entropy2 = 0
  for variable in variables:
      entropy = 0
      for target_variable in target_variables:
          num = len(df[attribute][df[attribute]==variable][df[target] ==target_variable])
          den = len(df[attribute][df[attribute]==variable])
          fraction = num/(den+eps)
          entropy += -fraction*log(fraction+eps)
      fraction2 = den/len(df)
      entropy2 += -fraction2*entropy
  return abs(entropy2)

def find_winner(df):
    IG = []
    for key in df.keys()[:-1]:
        IG.append(find_entropy(df)-average_information(df,key))
    return df.keys()[:-1][np.argmax(IG)]

```

```
def get_subtable(df, node,value):
  return df[df[node] == value].reset_index(drop=True)

"""# Build Decision Tree"""

def buildTree(df,tree=None):
    target = df.keys()[-1]   #target column

    #Here we build our decision tree

    #Get attribute with maximum information gain
    node = find_winner(df)

    #Get distinct value of that attribute e.g Salary is node and Low,Med and High are values
    attValue = np.unique(df[node])

    #Create an empty dictionary to create tree
    if tree is None:
        tree={}
        tree[node] = {}

    #We make loop to construct a tree by calling this function recursively.
    #In this we check if the subset is pure and stops if it is pure.

    for value in attValue:

        subtable = get_subtable(df,node,value)
        clValue,counts = np.unique(subtable[target],return_counts=True)

        if len(counts)==1:#Checking purity of subset
            tree[node][value] = clValue[0]
        else:
            tree[node][value] = buildTree(subtable) #Calling the function recursively

    return tree


"""# Reading the data set"""

df=pd.read_csv('https://raw.githubusercontent.com/ltdaovn/dataset/refs/heads/master/play_tennis.csv')
df = df.drop('day',axis=1)

# look at the number of rows and columns
print(f'Rows: {df.shape[0]}, Columns: {df.shape[1]}')

# look at the columns
print(df.columns)

"""df.info()"""

df.describe()

#build Tree
tree = buildTree(df)

import pprint
pprint.pprint(tree)
```