```python
import pandas as pd
import numpy as np
from math import log2

'''
Doing this by hand would have been a nightmare so I just wrote
code to do it because it makes life easier! Also I problably got
more out of writing code on how to do it rather than doing it by
hand because the true implementation of this would have been with
code.
'''


df = pd.read_csv('HW2 - Sheet1.csv')
df['RATING'] = df['RATING'].astype(str)

'''
Calculate Entropy for each attribute
'''
def entropy(attribute_labels):
    labels, counts = np.unique(attribute_labels, return_counts=True)
    probabilities = counts / counts.sum()
    return -sum(p * log2(p) for p in probabilities)

'''
Calculate infromation gain
data = dataframe
target = target column (WATCH)
attributes = list of attributes
'''
def information_gain(data, attribute, target):
    total_entropy = entropy(data[target])
    values, counts = np.unique(data[attribute], return_counts=True)
    weighted_entropy = sum((counts[i] / len(data)) * entropy(data.where(data[attribute] ==
values[i]).dropna()[target]) for i in range(len(values)))
    return total_entropy - weighted_entropy

'''
Recursively build the tree
data = dataframe
target = target column (WATCH)
attributes = list of attributes
'''
def build_tree(data, target, attributes):
    if len(np.unique(data[target])) <= 1:
        return np.unique(data[target])[0]
    elif len(attributes) == 0:
        return np.unique(data[target])[np.argmax(np.unique(data[target], return_counts=True)
[1])]
    else:
        best_attribute = max(attributes, key=lambda x: information_gain(data, x, target))
        tree = {best_attribute: {}}
        attributes.remove(best_attribute)
        for value in np.unique(data[best_attribute]):
            sub_data = data.where(data[best_attribute] == value).dropna()
            subtree = build_tree(sub_data, target, attributes)
            tree[best_attribute][value] = subtree
        return tree


attributes = list(df.columns)
attributes.remove('WATCH')

decision_tree = build_tree(df, 'WATCH', attributes)

'''
This function just displays the tree in a more readable format
```

```
'''
def print_tree(tree, indent=0):
    for key, value in tree.items():
        if isinstance(value, dict):
            print(" " * indent + f"|- {key}")
            print_tree(value, indent + 3)
        elif isinstance(value, str):
            print(" " * indent + f"|-  {key}: {value}")
        else:
            print(" " * indent + f"|- {key}:")
            print_tree(value, indent + 3)

print_tree(decision_tree)
print('\n')
print(decision_tree)

'''
------------------OUTPUT------------------

|- GENRE
   |- action
      |- PRICE
         |-  high: no
         |-  low: no
         |-  ok: yes
   |-  comedy: no
   |- drama
      |- RATING
         |- 3
            |- LENGTH
               |-  medium: no
               |-  very-long: no
         |-  4: maybe
         |-  5: maybe

Note that some values such as $$$ and 3-star are not in the tree because
they are not needed to make a decision. For example, there is no instance
where GENRE = action, and PRICE = $$$, so it is left out of the tree.
In my testing, if the value is left out of the tree, it is marked as "NO"


*************This is the tree in dictionary form*************
{'GENRE': {'action': {'PRICE': {'high': 'no', 'low': 'no', 'ok': 'yes'}}, 'comedy': 'no',
'drama': {'RATING': {'3': {'LENGTH': {'medium': 'no', 'very-long': 'no'}}, '4': 'maybe', '5':
'maybe'}}}}


------------------OUTPUT------------------
'''
```

GENRE	PRICE	LENGTH	RATING	WATCH
drama	$$$	short	5	yes
drama	ok	medium	4	maybe
drama	$$$	very-long	4	maybe
drama	$$$	medium	3	yes
drama	$$$	short	4	no
drama	$$$	medium	5	maybe
drama	$$$	medium	3	no
drama	low	medium	5	maybe
drama	$$$	medium	5	maybe
drama	ok	very-long	3	no
comedy	high	medium	5	no
comedy	ok	long	5	no
comedy	low	medium	5	no
comedy	high	long	5	no
comedy	$$$	medium	4	no
comedy	ok	medium	5	no
comedy	$$$	medium	4	no
action	ok	very-long	5	yes
action	ok	very-long	3	yes
action	low	long	5	no
action	low	long	3	no
action	high	long	3	no

Testing

Row	accuracy
1	X
2	√
3	√
4	√
5	√
6	X
7	X
8	√
9	X
10	X

If value not in my tree, I mark it NO

5/10

50 % accuracy

(trees don't generalize well)