pooya sharifi

shayan kashani

First we import the *numpy* library and read the dataset.

```
import numpy as np
import pandas as pd
```

first we need to find the keys in the data frame for example we have parent feature ,which could be usual or great parent ,we want to extract usual and ....

```
def find_entropy(df):
    Class = df.keys()[-1]   #To make the code generic, changing target variable class name
    entropy = 0
    values = df[Class].unique()
    print(values)
    for value in values:
        fraction = df[Class].value_counts()[value]/len(df[Class])
        entropy += -fraction*np.log2(fraction)
    return entropy
```

then we find the entropy for each feature

```
def find_entropy_attribute(df,attribute):
    Class = df.keys()[-1]
    target_variables = df[Class].unique()  #This gives all 'Yes' and 'No'
    variables = df[attribute].unique()
    entropy2 = 0
    for variable in variables:
        entropy = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute]==variable][df[Class] ==target_variable])
            den = len(df[attribute][df[attribute]==variable])
            fraction = num/(den+0.000000001)#eps
            entropy += -fraction*np.log2(fraction+0.000000001)#eps
        fraction2 = den/len(df)
        entropy2 += -fraction2*entropy
    return abs(entropy2)
```

we read the data set

```
import io
df2 = pd.read_csv('nursery.csv')

print("df2",df2)
data=pd.DataFrame(df2)
print("data",data)

print(find_entropy(df2))
print(find_entropy_attribute(df2,"parents"))
```

```
df2           parents   has_nurs      form children     housing     finance  \
0           usual     proper  complete        1  convenient  convenient
1           usual     proper  complete        1  convenient  convenient
2           usual     proper  complete        1  convenient  convenient
3           usual     proper  complete        1  convenient  convenient
4           usual     proper  complete        1  convenient  convenient
...           ...        ...       ...      ...         ...         ...
12955  great_pret  very_crit    foster     more    critical      inconv
12956  great_pret  very_crit    foster     more    critical      inconv
12957  great_pret  very_crit    foster     more    critical      inconv
12958  great_pret  very_crit    foster     more    critical      inconv
12959  great_pret  very_crit    foster     more    critical      inconv
```

```
              social       health final evaluation
0            nonprob  recommended        recommend
1            nonprob     priority         priority
2            nonprob    not_recom        not_recom
3      slightly_prob  recommended        recommend
4      slightly_prob     priority         priority
...              ...          ...              ...
12955  slightly_prob     priority       spec_prior
12956  slightly_prob    not_recom        not_recom
12957    problematic  recommended       spec_prior
12958    problematic     priority       spec_prior
12959    problematic    not_recom        not_recom

[12960 rows x 9 columns]
data           parents   has_nurs      form children     housing     finance  \
0           usual     proper  complete        1  convenient  convenient
1           usual     proper  complete        1  convenient  convenient
2           usual     proper  complete        1  convenient  convenient
3           usual     proper  complete        1  convenient  convenient
4           usual     proper  complete        1  convenient  convenient
...          ...        ...       ...      ...         ...         ...
12955  great_pret  very_crit    foster     more    critical      inconv
12956  great_pret  very_crit    foster     more    critical      inconv
12957  great_pret  very_crit    foster     more    critical      inconv
12958  great_pret  very_crit    foster     more    critical      inconv
12959  great_pret  very_crit    foster     more    critical      inconv

              social       health final evaluation
0            nonprob  recommended        recommend
1            nonprob     priority         priority
2            nonprob    not_recom        not_recom
3      slightly_prob  recommended        recommend
4      slightly_prob     priority         priority
...              ...          ...              ...
12955  slightly_prob     priority       spec_prior
12956  slightly_prob    not_recom        not_recom
12957    problematic  recommended       spec_prior
12958    problematic     priority       spec_prior
12959    problematic    not_recom        not_recom

[12960 rows x 9 columns]
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
1.7164959001837934
1.6435612869098675
```

- we create a function to find the highest information gain,
- the key which we enumerate on ,in basically the features of our data set
- in for:first we find the entropy of the data set,then we find the entropy of each feature and we minus it
- after the for is done we return the maximum argument(all atributes except the target)

```
def find_highest_info(df):
    Entropy_att = []
    IG = []
    for key in df.keys()[:-1]:#         Entropy_att.append(find_entropy_attribute(df,key))
        IG.append(find_entropy(df)-find_entropy_attribute(df,key))
    return df.keys()[:-1][np.argmax(IG)]
```

we test our find_highest_info function and we see that health has the highest information gain , so the first division should be based on health

```
print(find_highest_info(df2))
```

```
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior']
health
```

Double-click (or enter) to edit

it will only return the sub tree which has this attribute

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

we test to see wether we can get the subtree for health priority

```
print(get_subtable(df2,"health","priority"))
```

```
             parents   has_nurs      form children     housing     finance  \
0              usual     proper  complete        1  convenient  convenient   
1              usual     proper  complete        1  convenient  convenient   
2              usual     proper  complete        1  convenient  convenient   
3              usual     proper  complete        1  convenient      inconv   
4              usual     proper  complete        1  convenient      inconv   
...              ...        ...       ...      ...         ...         ...   
4315      great_pret  very_crit    foster     more    critical  convenient   
4316      great_pret  very_crit    foster     more    critical  convenient   
4317      great_pret  very_crit    foster     more    critical      inconv   
4318      great_pret  very_crit    foster     more    critical      inconv   
4319      great_pret  very_crit    foster     more    critical      inconv   

               social    health final evaluation  
0             nonprob  priority         priority  
1       slightly_prob  priority         priority  
2         problematic  priority         priority  
3             nonprob  priority         priority  
4       slightly_prob  priority         priority  
...               ...       ...              ...  
4315    slightly_prob  priority       spec_prior  
4316      problematic  priority       spec_prior  
4317          nonprob  priority       spec_prior  
4318    slightly_prob  priority       spec_prior  
4319      problematic  priority       spec_prior  

[4320 rows x 9 columns]
```

make tree

- count to see how many times we use this function recursively
- for the first node we get the attribute with maximum information gain
- Create an empty dictionary to create tree
- List item
- List item
- List item

```
def buildTree(df,tree=None):
    Class = df.keys()[-1]
    node = find_highest_info(df)
    attValue = np.unique(df[node])
    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 st
    for value in attValue:
        subtable = get_subtable(df,node,value)
        clValue,counts = np.unique(subtable['final evaluation'],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
    print("size of tree",counts)
    return tree
```

bulding the tree

```
df_train=df2.iloc[:8400,]
df_test=df2.iloc[8400:12959,:]
print("train",df_train)
print("test",df_test)
```

```
train           parents   has_nurs        form children     housing     finance  \
0           usual     proper    complete        1  convenient  convenient
1           usual     proper    complete        1  convenient  convenient
2           usual     proper    complete        1  convenient  convenient
3           usual     proper    complete        1  convenient  convenient
4           usual     proper    complete        1  convenient  convenient
...           ...        ...         ...      ...         ...         ...
8395  pretentious  very_crit  incomplete     more   less_conv  convenient
8396  pretentious  very_crit  incomplete     more   less_conv  convenient
8397  pretentious  very_crit  incomplete     more   less_conv      inconv
8398  pretentious  very_crit  incomplete     more   less_conv      inconv
8399  pretentious  very_crit  incomplete     more   less_conv      inconv

             social       health final evaluation
0           nonprob  recommended        recommend
1           nonprob     priority         priority
2           nonprob    not_recom        not_recom
3     slightly_prob  recommended        recommend
4     slightly_prob     priority         priority
...             ...          ...              ...
8395    problematic     priority       spec_prior
8396    problematic    not_recom        not_recom
8397        nonprob  recommended       spec_prior
8398        nonprob     priority       spec_prior
8399        nonprob    not_recom        not_recom

[8400 rows x 9 columns]
test            parents   has_nurs        form children    housing finance  \
8400   pretentious  very_crit  incomplete     more  less_conv  inconv
8401   pretentious  very_crit  incomplete     more  less_conv  inconv
8402   pretentious  very_crit  incomplete     more  less_conv  inconv
8403   pretentious  very_crit  incomplete     more  less_conv  inconv
8404   pretentious  very_crit  incomplete     more  less_conv  inconv
...            ...        ...         ...      ...        ...     ...
12954   great_pret  very_crit      foster     more   critical  inconv
12955   great_pret  very_crit      foster     more   critical  inconv
12956   great_pret  very_crit      foster     more   critical  inconv
12957   great_pret  very_crit      foster     more   critical  inconv
12958   great_pret  very_crit      foster     more   critical  inconv

              social       health final evaluation
8400   slightly_prob  recommended       spec_prior
8401   slightly_prob     priority       spec_prior
8402   slightly_prob    not_recom        not_recom
8403     problematic  recommended       spec_prior
8404     problematic     priority       spec_prior
...              ...          ...              ...
12954  slightly_prob  recommended       spec_prior
12955  slightly_prob     priority       spec_prior
12956  slightly_prob    not_recom        not_recom
12957    problematic  recommended       spec_prior
12958    problematic     priority       spec_prior

[4559 rows x 9 columns]
```

train

```
tree=buildTree(df_train)
print(tree)
```

```
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
size of tree [4]
size of tree [12]
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
size of tree [4]
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
['priority' 'spec_prior']
size of tree [4]
size of tree [13]
size of tree [20 35]
size of tree [ 62 103]
size of tree [126 370]
size of tree [1753    2  717  328]
```

✓ 9s completed at 10:53 PM