Gain_entropy

February 26, 2024

Se importan las librerias

[ ]:
```
import numpy as np
import pandas as pd
```

Se hace una función para obtener la entropia de los valores de class

[ ]:
```
def Entropy(x,y):
    if x == 0 or y == 0:
        return 0
    return -x/(x+y)*np.log2(x/(x+y))-y/(x+y)*np.log2(y/(x+y))
```

Se crea el dataframe

[ ]:
```
df = pd.DataFrame({'Age': ["young", 'young', 'young',␣
↪'young',"middle","middle","middle","middle","old","old","old","old","old"],
                   "Has_job":
↪[False,False,True,True,False,True,False,False,False,False,True,True,False],
                   "Own_house":
↪[False,False,False,True,False,True,True,True,True,True,False,False,False],
                   "Credit_Rating":
↪["fair","excellent","good","good","good","good","excellent","excellent","excellent","good",
                   "Class":␣
↪["no","no","yes","yes","no","yes","yes","yes","yes","yes","yes","no","yes"]})
```

En este caso se eliminaron los registros 5 y 6 de la base de datos original

[ ]:
```
df
```

[ ]:

	Age	Has_job	Own_house	Credit_Rating	Class
0	young	False	False	fair	no
1	young	False	False	excellent	no
2	young	True	False	good	yes
3	young	True	True	good	yes
4	middle	False	False	good	no
5	middle	True	True	good	yes
6	middle	False	True	excellent	yes
7	middle	False	True	excellent	yes
8	old	False	True	excellent	yes

```
9      old    False       True          good   yes
10     old     True      False          good   yes
11     old     True      False     excellent    no
12     old    False      False          fair   yes
```

Se usa value counts para obtener la cantidad de valores de la columna clase

[ ]:
```
df['Class'].value_counts()
```

[ ]:
```
yes    9
no     4
Name: Class, dtype: int64
```

Al usar entropy se obtiene la entropia de los valores de la columna clase

[ ]:
```
entropy = Entropy(9,4)
print(entropy)
```

```
0.8904916402194913
```

Se crea una función para obtener la entropia por cada variable de la columna y se aplica de manera sistemática

[ ]:
```
def entropy_attribute(x,y):
    return (x+y)/13*Entropy(x,y)
```

[ ]:
```
df["Class"].groupby(df["Age"]).value_counts()
```

[ ]:
```
Age     Class
middle  yes      3
        no       1
old     yes      4
        no       1
young   no       2
        yes      2
Name: Class, dtype: int64
```

[ ]:
```
middle = entropy_attribute(3,1)
old = entropy_attribute(4,1)
young= entropy_attribute(2,2)
Age_entropy = middle + old + young
print("young:",young, "middle:",middle, "old:",old)
print("Age entropy: ",Age_entropy)
age_gain = entropy - Age_entropy
print("Gain:",age_gain)
```

```
young: 0.3076923076923077 middle: 0.2496240382951178 old: 0.27766465187975475
Age entropy:  0.8349809978671803
Gain: 0.05551064235231107
```

[ ]:
```
df["Class"].groupby(df["Has_job"]).value_counts()
```

[ ]:
```
Has_job  Class
False    yes      5
         no       3
True     yes      4
         no       1
Name: Class, dtype: int64
```

[ ]:
```
Has_job_entropy = entropy_attribute(4,1)
dont_job_entropy = entropy_attribute(5,3)
Job_entropy = Has_job_entropy + dont_job_entropy
print("Has job:",Has_job_entropy, "Unemployed:",dont_job_entropy)
print("Job entropy: ",Job_entropy)
job_gain = entropy - Job_entropy
print("Gain:",job_gain)
```

```
Has job: 0.27766465187975475 Unemployed: 0.5873440017999785
Job entropy:  0.8650086536797332
Gain: 0.025482986539758112
```

[ ]:
```
df["Class"].groupby(df["Own_house"]).value_counts()
```

[ ]:
```
Own_house  Class
False      no       4
           yes      3
True       yes      6
Name: Class, dtype: int64
```

[ ]:
```
Has_jouse = entropy_attribute(6,0)
Homeless = entropy_attribute(3,4)
House_entropy= Has_jouse + Homeless
print("Own house:",Has_jouse, "Homeless:",Homeless)
print("House entropy: ",House_entropy)
house_gain = entropy - House_entropy
print("Gain:",house_gain)
```

```
Own house: 0.0 Homeless: 0.530507457864597
House entropy:  0.530507457864597
Gain: 0.3599841823548944
```

[ ]:
```
df["Class"].groupby(df["Credit_Rating"]).value_counts()
```

[ ]:
```
Credit_Rating  Class
excellent      yes      3
               no       2
fair           no       1
               yes      1
good           yes      5
```

```
                no       1
Name: Class, dtype: int64
```

[ ]:
```
excellent_entropy=entropy_attribute(3,2)
good_entropy=entropy_attribute(5,1)
fair_entropy=entropy_attribute(1,1)
credit_entropy=excellent_entropy+good_entropy+fair_entropy
print("Excellent:",excellent_entropy, "Good:",good_entropy, "Fair:
  ↪",fair_entropy)
print("Credit entropy: ",credit_entropy)
credit_gain = entropy - credit_entropy
print("Gain:",credit_gain)
```

```
Excellent: 0.3734425363287187 Good: 0.30001034845308655 Fair:
0.15384615384615385
Credit entropy:  0.8272990386279592
Gain: 0.06319260159153217
```

El mejor atributo para empezar sería el que tenga mejor ganancia de información, por esto se imprime cada valor para comparar

[ ]:
```
print("Age gain:",age_gain, "\nJob gain:",job_gain, "\nHouse gain:",house_gain,␣
  ↪"\nCredit gain:",credit_gain)
```

```
Age gain: 0.05551064235231107
Job gain: 0.025482986539758112
House gain: 0.3599841823548944
Credit gain: 0.06319260159153217
```

En este caso, el mejor atributo para empezar sería el atributo si tiene o no casa.