svxwcspsf

June 4, 2023

1 DEFINITION OF A METRIC

We start by loading the libraries we will use in this exercise.

[13]:
```
#!/usr/bin/python3

from sys import argv, exit

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

We load the dataset from the 'dataset.csv' file and then display the information from this dataset.

[14]:
```
dataframe = pd.read_csv("dataset.csv")
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            200 non-null    int64
 1   age                   200 non-null    float64
 2   height                200 non-null    float64
 3   job                   200 non-null    object
 4   city                  200 non-null    object
 5   favorite music style  200 non-null    object
dtypes: float64(2), int64(1), object(3)
memory usage: 9.5+ KB
```

We can see that there are 200 rows, 6 columns, no null values and different data types: Int, Float and Object.

We then display an example of the data within this dataset.

[15]:
```
dataframe.head(10)
```

[15]:
```
   Unnamed: 0        age      height         job       city  \
0           0  30.237071  179.874298    designer      paris
1           1  27.915796  172.659587     fireman  marseille
2           2  32.205338  181.337491     teacher      paris
3           3  26.595215  172.337885    designer   toulouse
4           4  27.394780  182.708030     teacher      paris
5           5  29.710022  177.037196      doctor   toulouse
6           6  41.789490  188.476525     fireman  marseille
7           7  36.665622  175.102745     painter   toulouse
8           8  33.838742  176.441942     painter      paris
9           9  30.037158  191.462290  developper      paris

  favorite music style
0                 trap
1               hiphop
2                metal
3                metal
4                metal
5                 rock
6                  rap
7                 rock
8            classical
9                 trap
```

We display all the unique values

[16]:
```
print(dataframe["favorite music style"].unique())
print(dataframe["city"].unique())
print(dataframe["job"].unique())
```

```
['trap' 'hiphop' 'metal' 'rock' 'rap' 'classical' 'other' 'jazz'
 'technical death metal']
['paris' 'marseille' 'toulouse' 'madrid' 'lille']
['designer' 'fireman' 'teacher' 'doctor' 'painter' 'developper' 'engineer']
```

We get to the special dissimilarity between cities by computing the distance between them.

There is the table representing distances between all cities

[17]:
```
cites = pd.DataFrame({
    'paris':    [0.0    , 0.661, 0.588, 2.0   , 0.217],
    'marseille':[0.661, 0.0    , 0.319, 2.0   , 1    ],
    'toulouse': [0.588, 0.319, 0.0    , 2.0   , 0.894],
    'madrid':   [2.0   , 2.0   , 2.0   , 0   , 2.0   ],
    'lille':    [0.217, 1    , 0.894, 2.0   , 0.0    ]
    })
cites
```

[17]:

	paris	marseille	toulouse	madrid	lille
0	0.000	0.661	0.588	2.0	0.217
1	0.661	0.000	0.319	2.0	1.000
2	0.588	0.319	0.000	2.0	0.894
3	2.000	2.000	2.000	0.0	2.000
4	0.217	1.000	0.894	2.0	0.000

This function will compute the distances between two cities (it will return 2 if one of the cities is Madrid because it is the city of another country).

[18]:
```
def get_city_idx(city):
    i = 0
    for city in cites:
        if city.lower() == city:
            break
        i += 1
    return i

def get_city_diff(city_1, city_2):
    return cites.loc[get_city_idx(city_1)][get_city_idx(city_2)]
```

“This function will compute the distance between two musical genres. If they are the same, it returns 0; otherwise, it returns 1. However, when comparing”Technical Death Metal” and “Metal,” it will return 0.5

[19]:
```
def get_music_diff(music_1, music_2):
    if music_1 == music_2:
        return 0
    if "metal" in music_1 and "metal" in music_2:
        return 0.5
    return 1
```

this function is a simple diff between jobs

[20]:
```
def get_job_diff(job_1, job_2):
    if job_1 == job_2:
        return 0
    return 1
```

This compute the distance between numbers

[21]:
```
def get_diff_number(num_1, num_2):
    return (math.sqrt((num_1 - num_2)**2))
```

We create a function to find the dissimilarity between two rows.

[22]:
```
def get_diff(p_1, p_2):
    age_diff = get_diff_number(dataframe.loc[p_1][1], dataframe.loc[p_2][1])
    height_diff = get_diff_number(dataframe.loc[p_1][2], dataframe.loc[p_2][2])
```

```
    job_diff = get_job_diff(dataframe.loc[p_1][3], dataframe.loc[p_2][3])
    city_diff = get_city_diff(dataframe.loc[p_1][4], dataframe.loc[p_2][4])
    music_diff = get_music_diff(dataframe.loc[p_1][5], dataframe.loc[p_2][5])
    return age_diff + height_diff + job_diff + city_diff + music_diff
```

We create the metric by calling the function we just created.

[23]:
```
nb_person = len(dataframe.index)
dissimilarity_matrix = np.zeros((nb_person, nb_person))
print("compute dissimilarities")
for p_1 in range(nb_person):
    for p_2 in range(nb_person):
        dissimilarity = get_diff(p_1, p_2)
        dissimilarity_matrix[p_1, p_2] = dissimilarity

print("dissimilarity matrix")
print(dissimilarity_matrix)
```

```
compute dissimilarities
dissimilarity matrix
[[ 0.         12.1969865   5.43145949 … 10.10580945 11.03302436
   7.53949478]
 [12.1969865   0.         15.62844599 …  4.09117704  3.40996214
  10.84059375]
 [ 5.43145949 15.62844599  0.         … 13.53726894 14.46448385
   6.03441996]
 …
 [10.10580945  4.09117704 13.53726894 …  0.          2.9272149
  10.0714167 ]
 [11.03302436  3.40996214 14.46448385 …  2.9272149   0.
  10.06863161]
 [ 7.53949478 10.84059375  6.03441996 … 10.0714167  10.06863161
   0.        ]]
```

[ ]:
```

```

And finaly we save the metric as ‘metric.npy’ file.

[24]:
```
np.save('metric.npy', dissimilarity_matrix)
exit(0)
```