II: Cleaning

January 30, 2022

0.0.1 Data Description

Research question

We are studying projects on the platform gofundme and the diverse factors that lead a project to be found. In our data set, we have extract the name, the date of the project, the description, the category, the pourcentage raised, the number of donors etc…

Here are the questions we want to answer : - Does the category have an influence on the founding ? - Does the pourcentage raised intuitively increase with time ? - Do Keywords in the description make a project more attractive ? - Does the donation per user influence the foundation ?

0.0.2 Useful imports

[33]:
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as mpl

pd.options.display.max_columns = 500
```

1 CLEANING THE DATAFRAME SCRAPED

[34]:
```
df=pd.read_csv('DataFinalfromWebScraping2.csv')
```

1.0.1 II.1/We transform Creation Date to a datetime

For this we will tokenize the str sentence Creation Date, create a dictionary for month and transform Created 2 days ago in 2022-01-14 for example with the function Convertdate.

Then, we transform it in a proper DateTime

[35]:
```
from datetime import date

#We use nltk to tokenize the str of Creation Date
import nltk
nltk.download('punkt')
```

```
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/albandhauthuille/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[36]:
```
def monthToNum(shortMonth):
    return {
            'January': 1,
            'February': 2,
            'March': 3,
            'April': 4,
            'May': 5,
            'June': 6,
            'July': 7,
            'August': 8,
            'September': 9,
            'October': 10,
            'November': 11,
            'December': 12
    }[shortMonth]
```

[37]:
```
def Convertdate():

    new_Date=[]
    #print(new_Date)
    for i in range (df['Creation Date'].size):
        #print(i)
        Creation_date = word_tokenize(df['Creation Date'][i])
        #print(Creation_date)

        if 'now'in Creation_date or 'hours' in Creation_date:
            new_Date.append(str(str(2022)+'-'+'01'+'-'+str(24)))
        elif 'ago' not in Creation_date:
            #print('no')
            new_Date.
↪append(str(Creation_date[4])+'-'+str(monthToNum(Creation_date[1]))+'-'+str(Creation_date[2]



        elif 'ago' in Creation_date:
            new_Date.
↪append(str(str(2022)+'-'+'01'+'-'+str(24-int(Creation_date[1]))))
    return new_Date
```

[38]:
```
df['Creation Date']=Convertdate()
```

[39]:
```
pd.to_datetime(df['Creation Date'])
df['Creation Date']=pd.to_datetime(df['Creation Date'])
```

1.0.2 II.2/ Retiring the '%' of Pourcentage Raised to plot properly

[40]:
```
pd.options.mode.chained_assignment = None  # default='warn'
for i in range(len(df['Pourcentage Raised'])):
    df['Pourcentage Raised'][i]=df['Pourcentage Raised'][i][:-1]
```

[41]:
```
#For example:
df['Pourcentage Raised'][4]
```

[41]:
```
'1.0'
```

1.0.3 II.3/ Convert to proper type

[42]:
```
df=df.convert_dtypes()
```

[43]:
```
df['Pourcentage Raised']=df['Pourcentage Raised'].astype(float)
```

1.0.4 II.4/ Duration of a collect with 'Creation Date'

[44]:
```
Duration =[]
for i in range(len(df['Creation Date'])):
    Duration.append((pd.Timestamp.today()-df['Creation Date'][i]).days)
df["Duration in days"]=Duration
```

1.0.5 II.5/ We only study the keywords, we reduce the 'Description' to 'Short_description' with words of length >4

[45]:
```
Short_description=[]
for i in range(len(df['Description'])):
    list=[]
    a=word_tokenize(df['Description'][i])
    for word in a:
        if len(word)>4:
            list.append(word)
    Short_description.append(list)

df['Short_description']=Short_description
#df
```

1.0.6 II.6.1/ Creation of column 'Amount Collected'

[46]:
```
Amount_Collected=[]
#print(df)
for i in range(len(df['collect'])):
    #word=(df['collect'][i])[1:]
    word=word_tokenize(df['collect'][i][1:])
    #If the amount is about millions:

    if 'M' in word[0]:
        Amount_Collected.append(float(word[0].replace(',', '.')[:-1])*1000000)
    #print(word)
    else:
        if ',' in word[0]:
            Amount_Collected.append(float(word[0].replace(',', '.'))*1000)
        else:
            Amount_Collected.append(float(word[0].replace(',', '.')))

df['Amount collected ']=Amount_Collected
```

1.0.7 II.6.2/ Creation of Column 'Amount targeted'

[47]:
```
Amount_Targeted=[]
for i in range(len(df['collect'])):
    word=word_tokenize(df['collect'][i])
    #print(word)
    #If the amount is about millions:
    if 'M' in word[-1]:
        Amount_Targeted.append(float(((word[-1].replace('M', '')).
 ↪replace('€','')).replace('£',''))*1000000)
    #print(word)
    elif 'B' in word[-1]:
        Amount_Targeted.append(float(((word[-1].replace('B', '')).
 ↪replace('€','')).replace('£',''))*1000000000)
    else:
        if ',' in word[-1]:
            Amount_Targeted.append(float(((word[-1].replace(',', '.')).
 ↪replace('€','')).replace('£',''))*1000)
        else:
            Amount_Targeted.append(float(((word[-1].replace(',', '.')).
 ↪replace('€','')).replace('£','')))

#len(Amount_Targeted)
df['Amount targeted']=Amount_Targeted
```

1.0.8 II.7/ MeanDonation

[48]:
```
MeanDonation =[]
for i in range(len(df['NumberDonors'])):
    if df['NumberDonors'][i]>0:
        MeanDonation.append((df['Amount collected '][i])/
 ↪(df['NumberDonors'][i]))
    else : MeanDonation.append(0)
df['MeanDonation']=MeanDonation
```

1.0.9 II.8/ Categories and conversion to numeric to plot and corr properly

[49]:
```
catégorie = ["Medical, Illness & Healing", "Funerals & Memorials", "Accidents &␣
 ↪Emergencies","Non-Profits & Charities","Education & Learning", "Animals &␣
 ↪Pets", "Environment", "Business & Entrepreneurs","Community & Neighbors",␣
 ↪"Competitions & Pageants","Creative Arts, Music & Film", "Celebrations &␣
 ↪Events","Missions, Faith & Church", "Babies, Kids & Family", "Sports, Teams␣
 ↪& Clubs", "Travel & Adventure","Volunteer & Service", "Dreams, Hopes &␣
 ↪Wishes", "Other"]
#print(catégorie)
```

[50]:
```
df['categoryNumeric'] = pd.factorize(df['Categorie'])[0] + 1
df['townNumeric'] = pd.factorize(df['town'])[0] + 1
```

1.0.10 II.9/ Creation of a column 'Size', which is the amount targeted:

[51]:
```
df.describe()
```

[51]:

	Unnamed: 0	Pourcentage Raised	NumberDonors	Duration in days	\
count	1728.000000	1728.00000	1728.000000	1728.00000	
mean	863.500000	22.10787	85.384259	61.33044	
std	498.974949	32.37733	1174.531989	69.87727	
min	0.000000	1.00000	0.000000	6.00000	
25%	431.750000	1.00000	0.000000	17.00000	
50%	863.500000	2.95000	3.000000	42.00000	
75%	1295.250000	30.45000	16.000000	73.00000	
max	1727.000000	100.00000	44392.000000	371.00000	

	Amount collected	Amount targeted	MeanDonation	categoryNumeric	\
count	1.728000e+03	1.728000e+03	1728.000000	1728.000000	
mean	7.450310e+03	3.493280e+06	39.468118	9.498264	
std	9.808869e+04	5.711015e+07	102.427454	5.191972	
min	0.000000e+00	1.000000e+00	0.000000	1.000000	
25%	0.000000e+00	1.000000e+03	0.000000	5.000000	
50%	9.000000e+01	3.000000e+03	20.000000	9.500000	
75%	6.887500e+02	1.000000e+04	46.666667	14.000000	
max	3.300000e+06	1.000000e+09	2950.962963	18.000000	

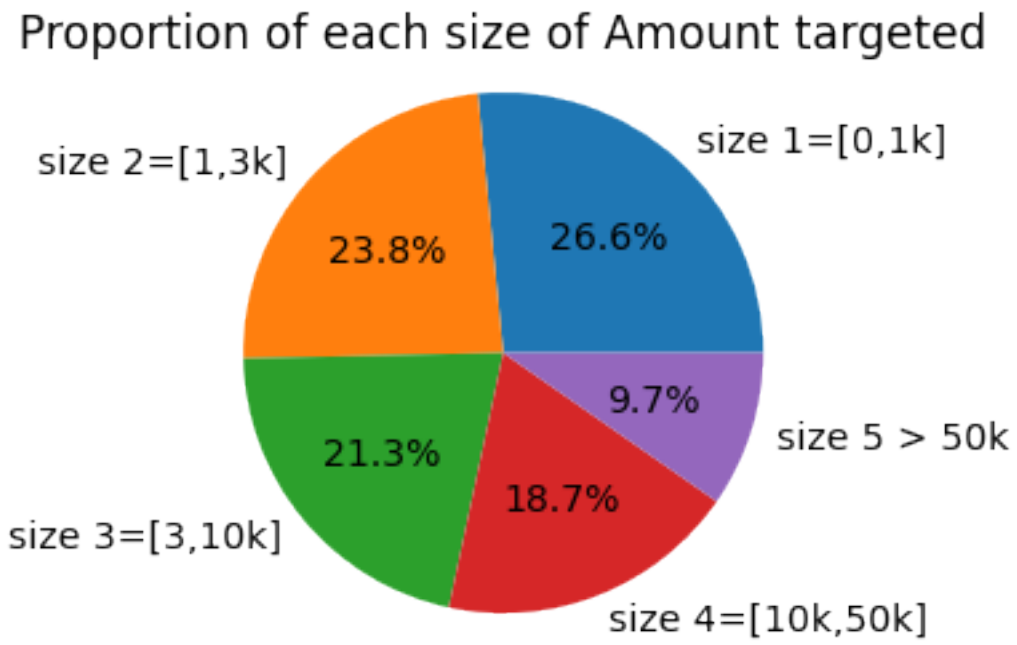
```
       townNumeric
count  1728.000000
mean    309.004051
std     278.631818
min       1.000000
25%      41.750000
50%     227.000000
75%     530.250000
max     911.000000
```

[52]:
```
size=[]
for collect in df['Amount targeted']:
    if collect < 1000:
        size.append(1)
    elif collect <3000:
        size.append(2)
    elif collect <10000:
        size.append(3)
    elif collect<50000:
        size.append(4)
    else:
        size.append(5)

df['size']=size
```

[53]:
```
fig1, ax1 = plt.subplots()
mpl.rcParams['font.size'] = 14.0
ax1.pie(df["size"].value_counts(),radius = 10, labels = ["size 1=[0,1k]", "size␣
↪2=[1,3k]", "size 3=[3,10k]" ,"size 4=[10k,50k]",'size 5 > 50k'], autopct='%1.
↪1f%%', textprops=dict(color="black"))
plt.title("Proportion of each size of Amount targeted", color="black")
ax1.axis('equal')
```

[53]:
```
(-11.055207994754944,
 11.002629001859779,
 -11.205109463409114,
 11.10265207950565)
```



1.0.11 II.10/ We drop useless columns

[54]:
```
df.drop(['Unnamed: 0', 'title', 'collect','Description'], axis=1, inplace=True)
```

1.0.12 II.11/ We drop the 4th quartile for eacht category which contains the highest, and sometimes fake, values

[55]:
```
df3=df.loc[df['Categorie']==catégorie[-1]]
df3=df3.loc[df3['Amount collected ']<df3['Amount collected '].describe()[6]]
for i in range(len(catégorie)-1,-1,-1):
    df2=df.loc[df['Categorie']==catégorie[i]]

    #drop 4th quartile of highest values
    df2=df2.loc[df2['Amount collected ']<df2['Amount collected '].describe()[6]]
    df3= pd.concat([df2,df3])
df=df3.reset_index(drop=True)
```

1.0.13 II.12/ Export cleaned DF for Analyse

[56]:
```
df.to_csv('datacleaned.csv')
```