IMPORTING PACKAGES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

LOAD DATA

In [2]:
```
raw_data = pd.read_csv('C:/Users/ronin/Google Drive/BUTTERFLIES/archive/okcupid_prof
```

In [3]:
```
okcupid_profiles = raw_data.iloc[:5002,0:21] # Ignore all the "essay" text columns f
```

EDA

In [4]:
```
# print(okcupid_profiles.shape)
total_nrows = okcupid_profiles.shape[0]
total_ncols = okcupid_profiles.shape[1]
print('Total rows: ', total_nrows, ' Total columns: ', total_ncols)
print(okcupid_profiles.info())
```

```
Total rows:  5002  Total columns:  21
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5002 entries, 0 to 5001
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          5002 non-null   int64
 1   status       5002 non-null   object
 2   sex          5002 non-null   object
 3   orientation  5002 non-null   object
 4   body_type    4543 non-null   object
 5   diet         3037 non-null   object
 6   drinks       4746 non-null   object
 7   drugs        3804 non-null   object
 8   education    4482 non-null   object
 9   ethnicity    4508 non-null   object
 10  height       5002 non-null   float64
 11  income       5002 non-null   int64
 12  job          4304 non-null   object
 13  last_online  5002 non-null   object
 14  location     5002 non-null   object
 15  offspring    2060 non-null   object
 16  pets         3306 non-null   object
 17  religion     3296 non-null   object
 18  sign         4102 non-null   object
 19  smokes       4504 non-null   object
 20  speaks       4998 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 820.8+ KB
None
```

In [5]:
```
# append IDs as another column
df = okcupid_profiles
df.insert(0, "id", df.index +1001 , True)  # 1001 as starting id
okcupid_profiles.head(2) #display first 2 entries
```

Out[5]:

	id	age	status	sex	orientation	body_type	diet	drinks	drugs	education	..
0	1001	22	single	m	straight	a little extra	strictly anything	socially	never	working on college/university	.
1	1002	35	single	m	straight	average	mostly other	often	sometimes	working on space camp	.

2 rows × 22 columns

In [6]:

```
print(okcupid_profiles.isna().sum())
for col in okcupid_profiles.drop(['id', 'age'],axis=1).columns:
    print(f"-----------{col}-----------")
    print(okcupid_profiles[col].nunique())
```

```
id                0
age               0
status            0
sex               0
orientation       0
body_type       459
diet           1965
drinks          256
drugs          1198
education       520
ethnicity       494
height            0
income            0
job             698
last_online       0
location          0
offspring      2942
pets           1696
religion       1706
sign            900
smokes          498
speaks            4
dtype: int64
-----------status-----------
4
-----------sex-----------
2
-----------orientation-----------
3
-----------body_type-----------
12
-----------diet-----------
18
-----------drinks-----------
6
-----------drugs-----------
3
-----------education-----------
30
-----------ethnicity-----------
87
```

```
----------height----------
28
----------income----------
13
----------job----------
21
----------last_online----------
3925
----------location----------
73
----------offspring----------
15
----------pets----------
15
----------religion----------
45
----------sign----------
46
----------smokes----------
5
----------speaks----------
1131
```

In [7]:

```
# NORMALIZING RANGED FEATURES

column = 'age'
okcupid_profiles[column] = (okcupid_profiles[column] - okcupid_profiles[column].min(
column = 'height'
okcupid_profiles[column] = (okcupid_profiles[column] - okcupid_profiles[column].min(
okcupid_profiles.head(2)
```

Out[7]:

	id	age	status	sex	orientation	body_type	diet	drinks	drugs	educatic
0	1001	0.043478	single	m	straight	a little extra	strictly anything	socially	never	working c college/universi
1	1002	0.184783	single	m	straight	average	mostly other	often	sometimes	working on spac cam

2 rows × 22 columns

In [8]:

```
# okcupid_profiles[column].describe()
# import seaborn as sns
# sns.boxplot(x=okcupid_profiles[column])
# okcupid_profiles["height"].unique()
```

LOAD LIKES DATA

In [9]:

```
#load likes dataset
likes_data = pd.read_csv('C:/Users/ronin/Google Drive/BUTTERFLIES/likesdata.csv',ind
```

In [10]:

```
likes_data.head()
```

```
#print(likes_data.shape)
# likes_data.loc[1004,'1003']
```

Out[10]:

	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	...	5991	5992	5993	599
1001	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1002	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1004	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	1.0	...	NaN	NaN	NaN	NaN
1005	NaN	NaN	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

5 rows × 5000 columns

In [11]:

```
#FOR TESTING, FINDING THE LIST OF LIKED PROFILES FOR AN ID
listi = [] #list of liked profiles for some specified id

id = 1005 #id to be searched matches for (index)
for i in range(likes_data.shape[1]):
    if likes_data.loc[id][i]==1.0 :
        listi.append(likes_data.columns[i])

liked_list = listi
listi
```

Out[11]:

```
['1003', '1004', '1050', '1076', '1080']
```

In [12]:

```
#one hot encoding ie creates more columns for  1 0 mapping

dummydat2= okcupid_profiles[["id", "age","height", "status", "sex", "drinks", "smoke
nominal_features = pd.get_dummies(data = dummydat2,columns= ["status", "sex", "drink
nominal_features
```

Out[12]:

	id	age	height	status_married	status_seeing someone	status_single	sex_m	drinks_not at all	drin
0	1001	0.043478	0.487179	0	0	1	1	0	
1	1002	0.184783	0.358974	0	0	1	1	0	
2	1003	0.217391	0.307692	0	0	0	1	0	
3	1004	0.054348	0.384615	0	0	1	1	0	
4	1005	0.119565	0.256410	0	0	1	1	0	
...	...	...	...	...	...	...	...	...	
4997	5998	0.065217	0.307692	0	0	1	1	0	
4998	5999	0.108696	0.282051	0	0	1	0	0	
4999	6000	0.086957	0.153846	0	0	1	0	0	
5000	6001	0.260870	0.179487	0	0	1	0	0	
5001	6002	0.358696	0.256410	0	0	1	1	0	

5002 rows × 45 columns

LOAD OUTPUT CSV ie for RECOMMENDED PROFILES

In [13]:
```
# CSV TO RECOMMENDATION
#Load reco dataset
reco_data = pd.read_csv('C:/Users/ronin/Google Drive/BUTTERFLIES/Reco_data_output.cs
```

FINDING COSINE SIMILARITY

In [14]:
```
from sklearn.metrics.pairwise import cosine_similarity
from heapq import nlargest

def findmatches(curr):  # FUNCTION TO RETURN LIST OF RECO IDS CORRESPONDING TO CURR
    Dict = {}

#   curr = int(liked_list[1])
    curterm = nominal_features[nominal_features["id"]==curr]  #PICKOUT THE WHOLE ROW
    curterm.drop(labels="id", axis=1)   #remove id COLUMN before cosine similarty

    for t in nominal_features.id:   #ITERATE FOR ALL PROFILES(IDS)
        iterm = nominal_features[nominal_features["id"]==int(t)]  #ROW OF EACH ID
        iterm.drop(labels="id", axis=1)
        Dict[int(t)] = cosine_similarity(curterm,iterm )    #STORE THE KEY-ID AND VA


    best_matches = nlargest(3, Dict, key = Dict.get)       #PICK OUT TOP 3 MATCHES O
    return best_matches

# returns best match recommendation to curr
```

In [15]:
```
for i in range(1002,1500):   # RANGE TO WHICH RECO IDS ARE TO BE FOUND

#   listi = []
    id = i #id i to be searched matches for (index)
    matches = []

    for p in likes_data.columns: # TO FIND MATCHES OF i ID (likesdata(i,p) = 1 IF MA
        t=int(p)
        if likes_data.loc[id,str(t)]==1.0 :
#           listi.append(likes_data.columns[j])
            matches+=findmatches(t)  # IF A MATCH, FIND SIMILAR PROFILES TO MATCH PR


    for k in matches:
        reco_data.loc[i,str(k)]=1.0   # NOW MAKE CHANGES TO RECO CSV BY PUTTING 1 TO
```

In [18]:
```
#  reco_data.loc[1005][reco_data.loc[1005].notna()]  #list of recommendation for ID
# reco_data.head()
# reco_data.loc[1004,1010]
# reco_data.at[1004,'1009' ]
reco_data.head()

# matches1005 = reco_data.index[reco_data.loc[1005].notna()==True].to_list() # to pr
```

```
# nominal_features[nominal_features["id"].isin(matches1005)]

# similarto1003 = findmatches(1003)  #find similar profs of given id
# similarto1003
```

Out[18]:

	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	...	5991	5992	5993	599
1001	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1002	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1004	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	1.0	...	NaN	NaN	NaN	NaN
1005	NaN	NaN	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

5 rows × 5000 columns

In [17]:

```
reco_data.to_csv('resultrecos.csv',index=True)
```

In [19]:

```
matches1005 = reco_data.index[reco_data.loc[1005].notna()==True].to_list() # to prin
nominal_features[nominal_features["id"].isin(matches1005)]
```

Out[19]:

	id	age	height	status_married	status_seeing someone	status_single	sex_m	drinks_not at all	drin
2	1003	0.217391	0.307692	0	0	0	1	0	
3	1004	0.054348	0.384615	0	0	1	1	0	
49	1050	0.086957	0.256410	0	0	1	0	0	
54	1055	0.163043	0.333333	0	0	1	1	0	
61	1062	0.184783	0.487179	0	0	1	1	0	
71	1072	0.163043	0.282051	0	0	1	0	0	
75	1076	0.119565	0.282051	0	0	0	0	0	
79	1080	0.032609	0.384615	0	0	1	1	0	
163	1164	0.206522	0.256410	0	0	1	0	0	
171	1172	0.141304	0.358974	0	0	1	1	0	
421	1422	0.304348	0.384615	0	0	1	1	0	
444	1445	0.228261	0.358974	0	0	1	1	0	
482	1483	0.119565	0.384615	0	0	1	1	0	
483	1484	0.097826	0.153846	0	0	1	0	0	
1174	2175	0.195652	0.205128	0	0	1	0	0	

15 rows × 45 columns

In [20]:

```
similarto1003 = findmatches(1003)  #find similar profs of given id
similarto1003
```

Out[20]: [1003, 1445, 1483]

In [ ]: