Problem description

In an era where technology plays a significant role in people's lives, one cannot deny that it changes the way people interact and communicate with others. Today, technology has caused some significant changes in the dating world as well. Online dating is a new trend that is influencing many people around the world.

As a data scientist, you are required to predict the match percentage between the users in a matrix format based on the attributes provided by the user on a dating website.

Column name Description

user_id: Represents unique user IDs

username: Represents the name of a user

age: Represents the age of a user

status: Represents the relationship status of a user (Single, available, and so on)

sex: Represents the gender of a user

orientation: Represents the sexual orientation of a user (gay, bisexual, or straight)

drinks: Represents if a user likes to drink or not

drugs: Represents if a user consumes drugs or not

height: Represents the height of a user in inches

job: Represents the profession that a user

location: Represents where a user resides

pets: Represents if a user likes pets or not

smokes: Represents if a user smokes or not

language: Represents the languages spoken by a user

new_languages: Represents if a user is interested to learn a new language

body_profile: Represents the type of body a user has

education_level: Represents the educational level of a user

dropped_out: Represents if a user dropped out of school or college

bio: Represents a user's description

interests: Represents the interests of a user

other_interests: Represents other interests of a user

location_preference: Represents the preferred location to find a date

In [ ]:

```
```

In [2]:

```
!pip install gensim
```

```
Collecting gensim
  Downloading gensim-3.8.3-cp38-cp38-macosx_10_9_x86_64.whl (24.2 MB)
     |████████████████████████████████| 24.2 MB 3.2 MB/s eta 0:00:01
Requirement already satisfied: six>=1.5.0 in /Users/gewoorkar/opt/anaconda3/lib/python3.8/site-
packages (from gensim) (1.14.0)
```

```
Collecting smart-open>=1.8.1
  Downloading smart_open-4.0.1.tar.gz (117 kB)
     |████████████████████████████████| 117 kB 4.1 MB/s eta 0:00:01
Requirement already satisfied: scipy>=0.18.1 in /Users/gewoorkar/opt/anaconda3/lib/python3.8/site-
packages (from gensim) (1.4.1)
Requirement already satisfied: numpy>=1.11.3 in /Users/gewoorkar/opt/anaconda3/lib/python3.8/site-
packages (from gensim) (1.18.1)
Building wheels for collected packages: smart-open
  Building wheel for smart-open (setup.py) ... done
  Created wheel for smart-open: filename=smart_open-4.0.1-py3-none-any.whl size=108249
sha256=c0b25bc8ba1391dcf0d3c0340185654d7de9da8fb69854969fe1caec23d16269
  Stored in directory:
/Users/gewoorkar/Library/Caches/pip/wheels/8c/f9/f4/4ddd9ddee3488f48be20e9bf3108961f03ae23da29b7ed2

Successfully built smart-open
Installing collected packages: smart-open, gensim
Successfully installed gensim-3.8.3 smart-open-4.0.1
```

In [2]:

```
#importing the libraries
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import re
from nltk.corpus import stopwords
import nltk
from nltk.tokenize import word_tokenize
from scipy.sparse import csr_matrix
from tqdm import tqdm
from sklearn.preprocessing import StandardScaler
import pickle
from scipy.sparse import hstack

from sklearn.metrics.pairwise import cosine_similarity
%matplotlib inline
```

In [16]:

```
import gensim.downloader as api

w2v = api.load('word2vec-google-news-300')
```

1. Data overview

In [3]:

```
df = pd.read_csv('data.csv')
```

In [4]:

```
df
```

Out[4]:

	user_id	username	age	status	sex	orientation	drinks	drugs	height	job	...	smokes	language
0	fffe3100	Edith Lopez	27	single	f	gay	socially	never	66.0	medicine / health	...	no	english (fluently), spanish (poorly), sign lan...
													english

	user_id	username	age	status	sex	orientation	drinks	drugs	height	job	...	smokes	language
1	fffe3200	Travis Young	26	single	m	gay	socially	never	68.0	other	...	no	english (fluently), tagalog (okay), french (po...
2	fffe3300	Agnes Smith	20	seeing someone	f	bisexual	socially	sometimes	69.0	other	...	sometimes	english (fluently), sign language (poorly), fr...
3	fffe3400	Salvador Klaver	27	single	m	bisexual	socially	sometimes	68.0	computer / hardware / software	...	no	english
4	fffe3500	Elana Sewell	22	single	f	bisexual	often	sometimes	68.0	other	...	yes	english
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1996	fffe3100390039003700	Reynaldo Ellis	24	single	m	straight	socially	never	69.0	student	...	yes	english
1997	fffe3100390039003800	Laura Adams	23	single	f	gay	socially	never	68.0	sales / marketing / biz dev	...	no	english
1998	fffe3100390039003900	Daniel Duran	28	single	m	straight	rarely	never	71.0	sales / marketing / biz dev	...	no	english
1999	fffe3200300030003000	Robert Orr	32	seeing someone	m	straight	not at all	never	68.0	other	...	no	english (fluently), japanese (okay)
2000	fffe3200300030003100	Mildred Harwell	41	single	f	straight	socially	never	67.0	artistic / musical / writer	...	no	english (fluently), spanish (okay)

2001 rows × 22 columns

In [5]:

```
df.shape
```

Out[5]:

```
(2001, 22)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['user_id', 'username', 'age', 'status', 'sex', 'orientation', 'drinks',
       'drugs', 'height', 'job', 'location', 'pets', 'smokes', 'language',
       'new_languages', 'body_profile', 'education_level', 'dropped_out',
       'bio', 'interests', 'other_interests', 'location_preference'],
      dtype='object')
```

1.1 Checking for NULL values

In [6]:

```
print("No of NULL values : ", sum(df.isnull().any()))
```

```
No of NULL values :  0
```

1.2 Handling duplicate users

In [7]:

```
duplicates = df.duplicated()
print("There are {} duplicate users in the data..".format(sum(duplicates)))
```

```
There are 0 duplicate users in the data..
```

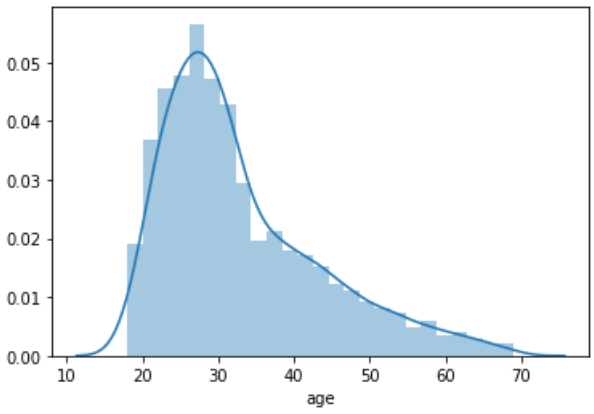
2. Exploratory Data Analysis

Let's try to visualise some of the features which seem relevant

2.1 'age'

In [8]:

```
#univariate analysis of age
sns.distplot(df['age'])
plt.show()
print('='*50)
print(df['age'].describe())
```

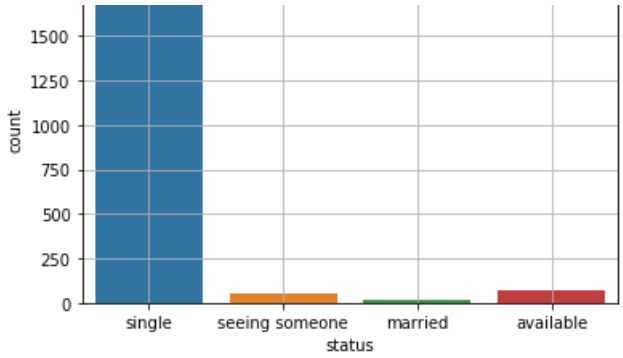



```
==================================================
count    2001.000000
mean       33.072464
std        10.483189
min        18.000000
25%        26.000000
50%        30.000000
75%        39.000000
max        69.000000
Name: age, dtype: float64
```

2.2 'status'

In [9]:

```
#univariate analysis on 'status'
sns.countplot(df['status'])
plt.grid()
plt.show()

print('='*50)
print(df['status'].describe())
```

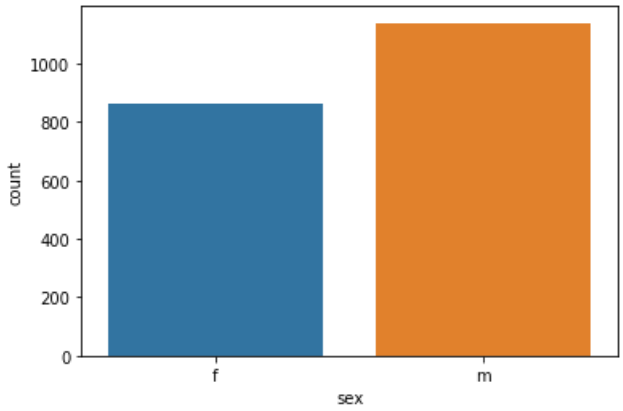



```
==================================================
count        2001
unique          4
top        single
freq         1867
Name: status, dtype: object
```

2.3 'sex'

In [10]:

```
#univariate analysis on 'sex'
sns.countplot(df['sex'])
plt.show()

print('='*50)
print(df['sex'].describe())
```

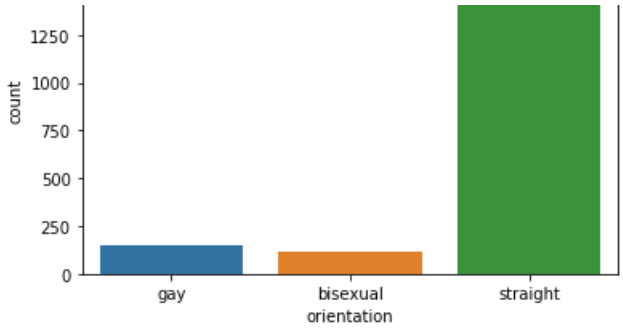



```
==================================================
count     2001
unique       2
top          m
freq      1139
Name: sex, dtype: object
```

2.4 'orientation'

In [11]:

```
#univariate analysis on 'orientation'
sns.countplot(df['orientation'])
plt.show()

print('='*50)
print(df['orientation'].describe())
```

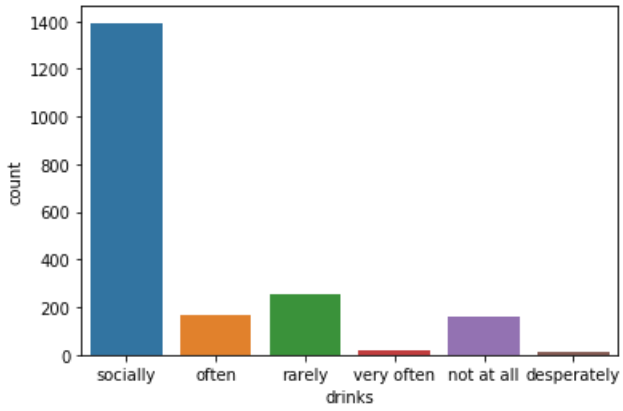



```
================================================
count          2001
unique            3
top        straight
freq           1736
Name: orientation, dtype: object
```

2.5 'drinks'

In [12]:

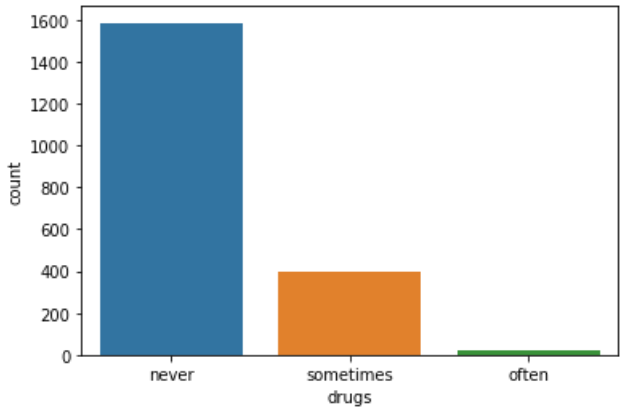
```
#univariate analysis on 'drinks'
sns.countplot(df['drinks'])
plt.show()
```



2.6 'drugs'

In [13]:

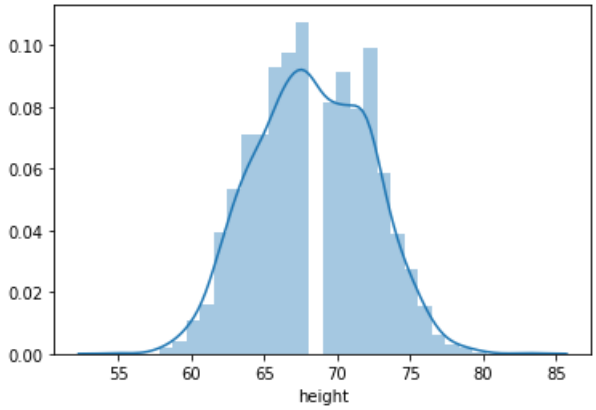
```
#univariate analysis on 'drugs'
sns.countplot(df['drugs'])
plt.show()
```



2.7 'height'

In [14]:

```
#univariate analysis on 'height'
sns.distplot(df['height'])
plt.show()
print('='*50)
print(df['height'].describe())
```

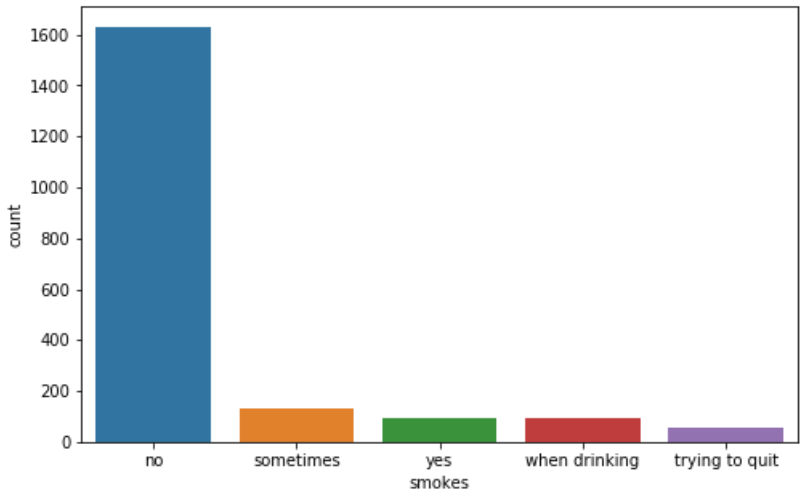



```
==================================================
count    2001.000000
mean       68.289855
std         3.895246
min        55.000000
25%        65.000000
50%        68.000000
75%        71.000000
max        83.000000
Name: height, dtype: float64
```

2.8 'smokes'

In [15]:

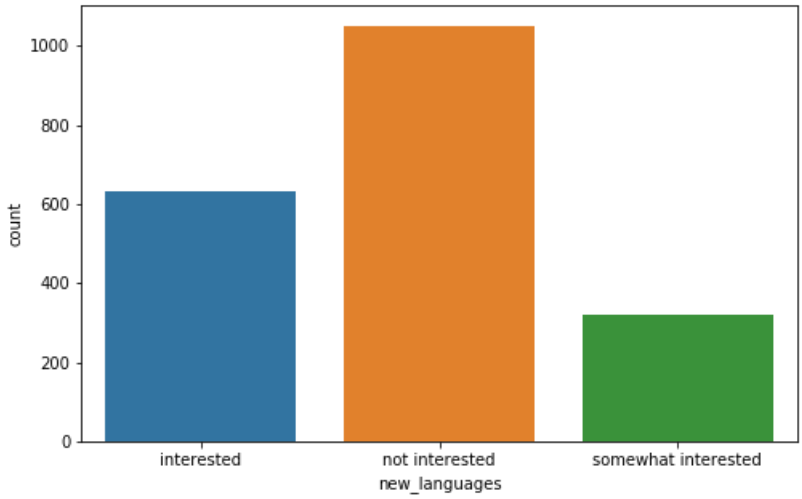
```
#univariate analysis on 'smokes'
plt.figure(figsize = (8,5))
sns.countplot(df['smokes'])
plt.show()
```



2.9 'new_languges'

In [16]:

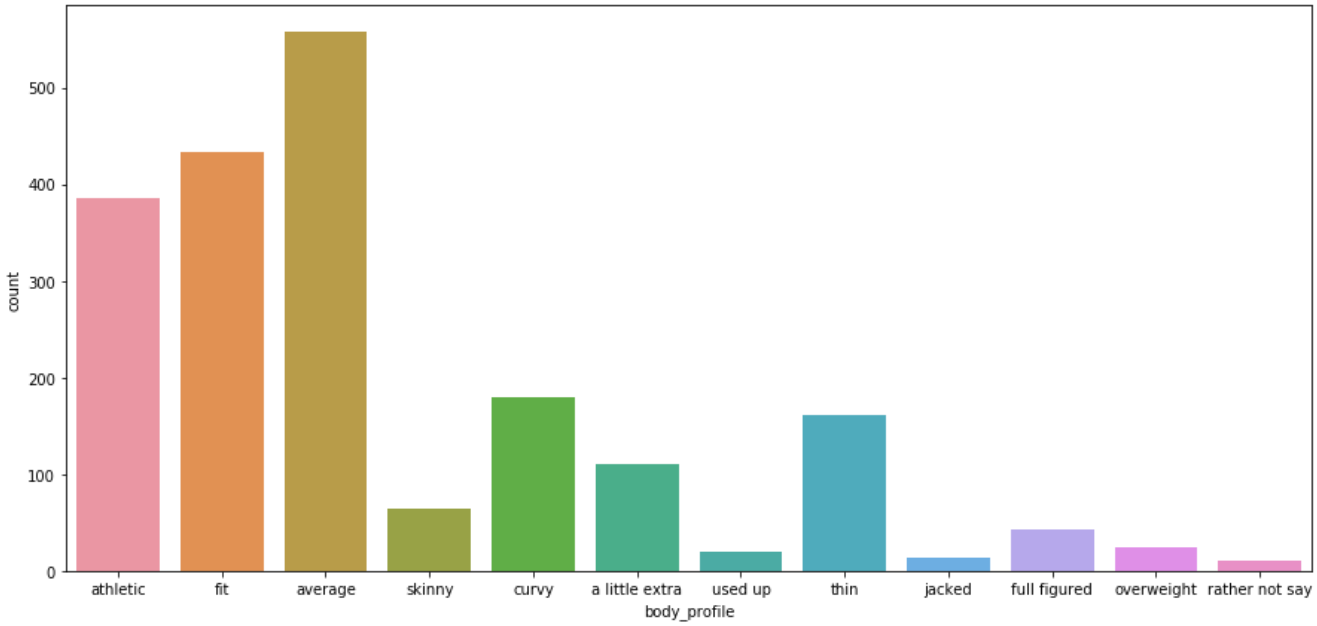
```
plt.figure(figsize = (8,5))
sns.countplot(df['new_languages'])
plt.show()
```



2.10 'body_profile'

In [17]:

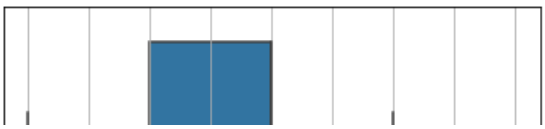
```
plt.figure(figsize = (15,7))
sns.countplot(df['body_profile'])
plt.show()
```

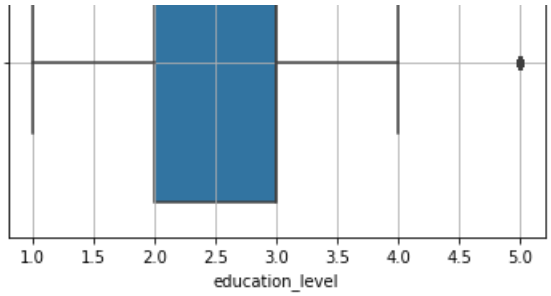


2.11 'education_level'

In [18]:

```
sns.boxplot(x = df['education_level'])
plt.grid()
plt.show()
print('='*50)
print(df['education_level'].describe())
```



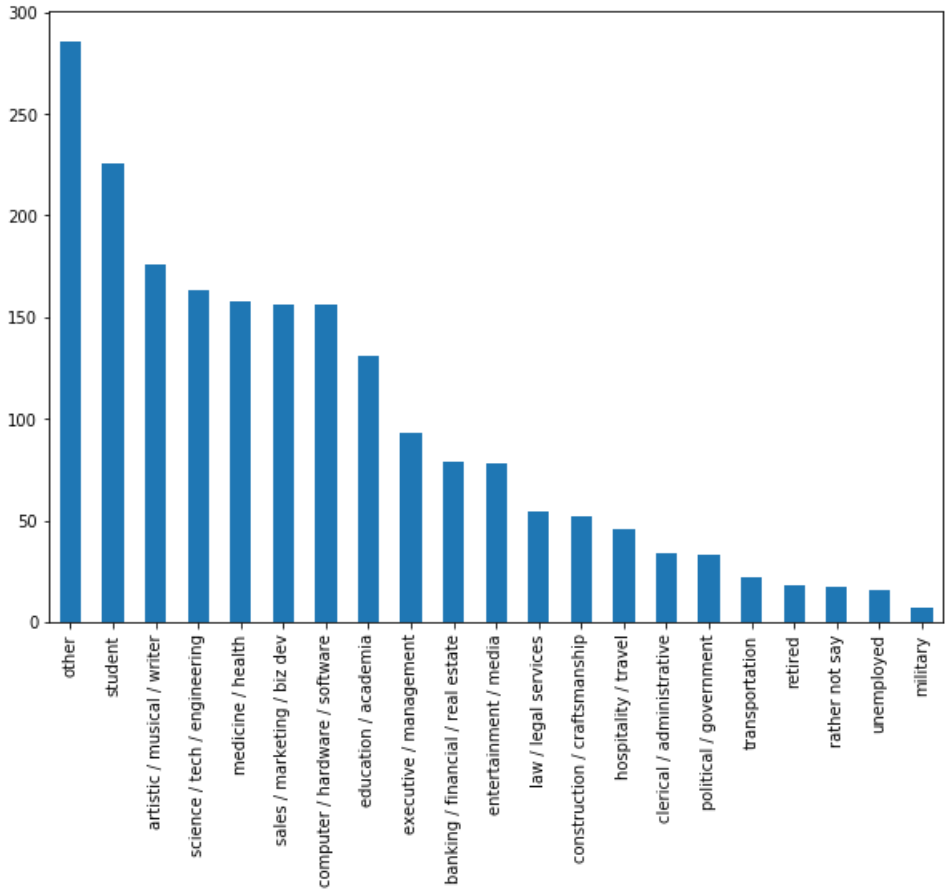



```
==================================================
count    2001.000000
mean        2.932534
std         0.812371
min         1.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         5.000000
Name: education_level, dtype: float64
```

2.12 'job'

In [19]:

```
plt.figure(figsize = (10,7))
df['job'].value_counts().plot(kind = 'bar')
plt.show()
```



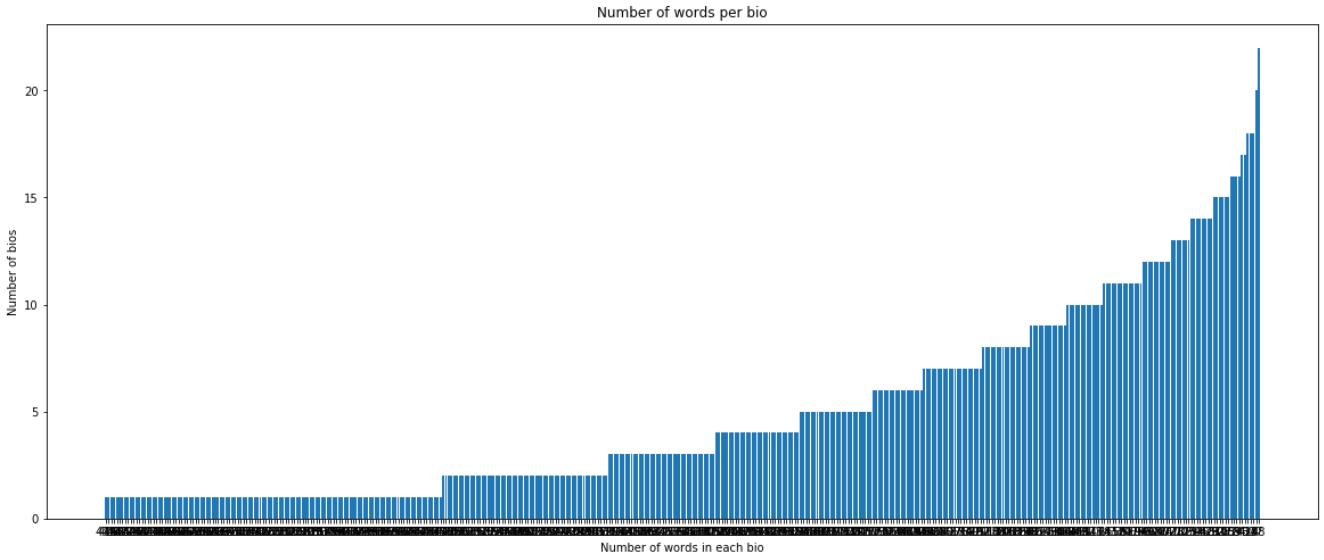
In [ ]:

```

```

2.13 'bio'

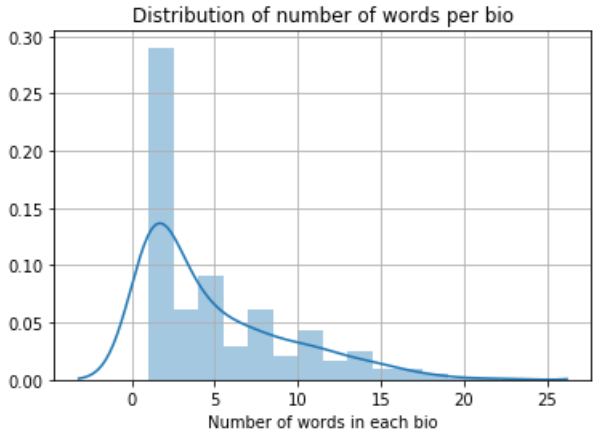
In [20]:

```
word_count = df['bio'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


ind = np.arange(len(word_dict))
plt.figure(figsize=(20,8))
p1 = plt.bar(ind, list(word_dict.values()))

plt.ylabel('Number of bios')
plt.xlabel('Number of words in each bio')
plt.title('Number of words per bio')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```



In [21]:

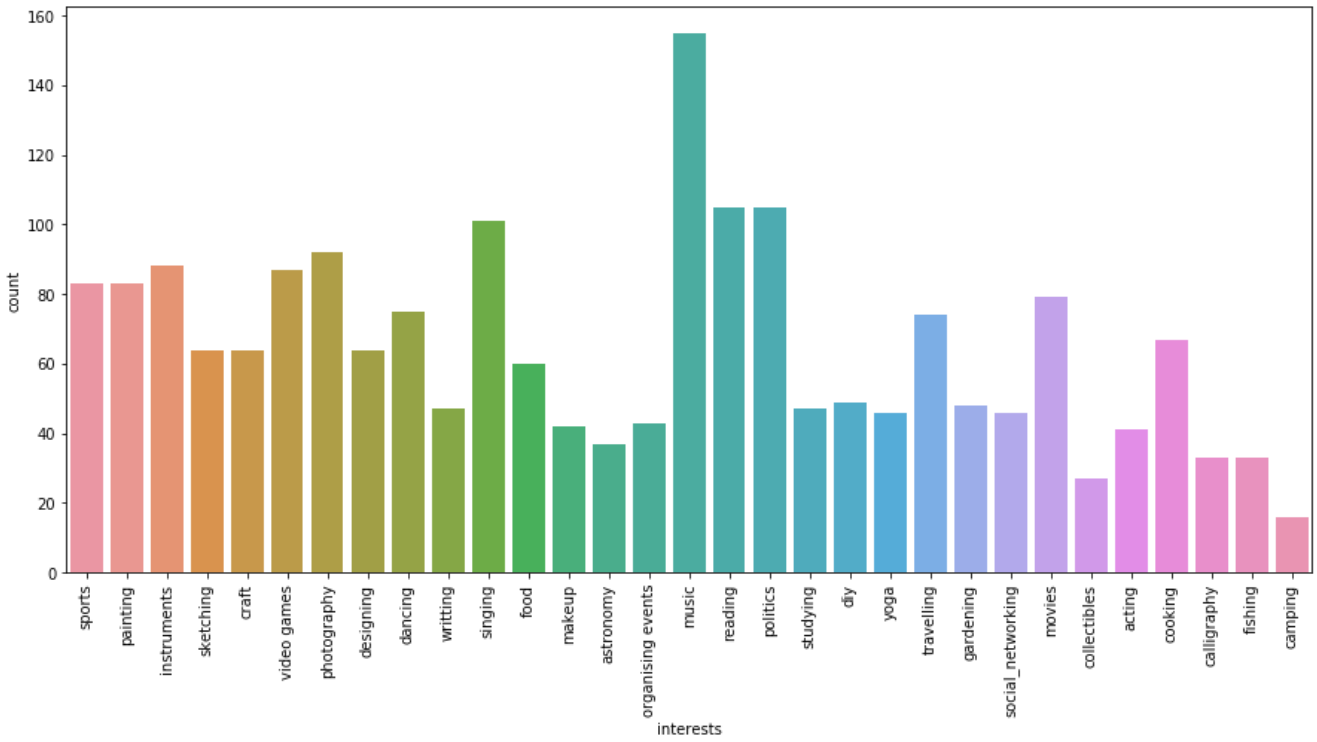
```
sns.distplot(word_count.values)
plt.grid()
plt.title('Distribution of number of words per bio')
plt.xlabel('Number of words in each bio')
plt.show()
```



2.14 'interests'

In [22]:

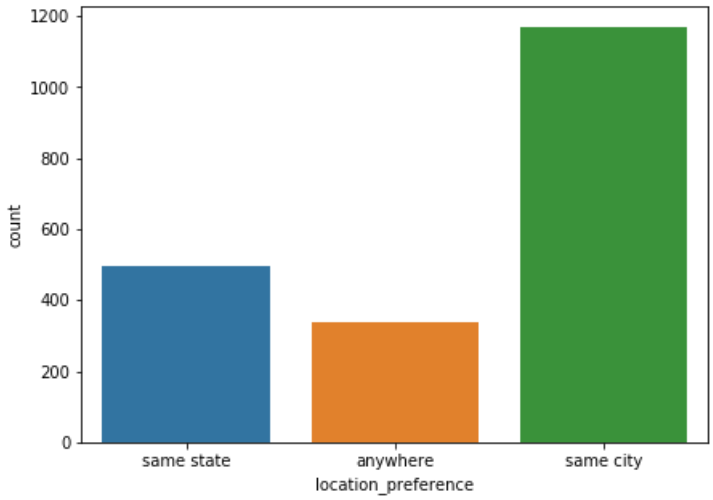
```
plt.figure(figsize = (15,7))
sns.countplot(df['interests'])
plt.xticks(rotation = 90)
plt.show()
```



2.15 'location_preference'

In [23]:

```
plt.figure(figsize = (7,5))
sns.countplot(df['location_preference'])
plt.show()
```



3. Data preprocessing

3.1 Preprocessing Categorical features

3.1.1 'job'

In [24]:

```
#removing the '/' and whitespaces
df['job'] = df['job'].str.replace('/','_')
df['job'] = df['job'].str.replace(' ','')
```

```
df['job'].value_counts()
```

Out[24]:

```
other                          286
student                        226
artistic_musical_writer        176
science_tech_engineering       163
medicine_health                158
sales_marketing_bizdev         156
computer_hardware_software     156
education_academia             131
executive_management            93
banking_financial_realestate    79
entertainment_media             78
law_legalservices               54
construction_craftsmanship      52
hospitality_travel              46
clerical_administrative         34
political_government            33
transportation                  22
retired                         18
rathernotsay                    17
unemployed                      16
military                         7
Name: job, dtype: int64
```

3.1.2 'location'

In [25]:

```
df['location']
```

Out[25]:

```
0             oakland, california
1       pleasant hill, california
2             oakland, california
3           daly city, california
4             oakland, california
                   ...
1996          oakland, california
1997    san francisco, california
1998    pleasant hill, california
1999    san francisco, california
2000          oakland, california
Name: location, Length: 2001, dtype: object
```

In [26]:

```
#removing ',' and whitespaces
df['location'] = df['location'].str.replace(',', ' ')
df['location'].value_counts()
```

Out[26]:

```
san francisco  california      911
oakland  california            250
berkeley  california           146
san mateo  california            56
palo alto  california            45
                               ...
foster city  california          1
point richmond  california       1
petaluma  california             1
canyon country  california       1
north hollywood  california      1
Name: location, Length: 70, dtype: int64
```

3.1.3 'pets'

In [27]:

```
#removing whitespaces
df['pets'] = df['pets'].str.replace(' ', '_')
df['pets'].value_counts()
```

Out[27]:

```
likes_dogs_and_likes_cats          743
likes_dogs                         303
likes_dogs_and_has_cats            229
has_dogs                           184
has_dogs_and_likes_cats            148
likes_dogs_and_dislikes_cats       121
has_dogs_and_has_cats               87
has_cats                            63
likes_cats                          59
has_dogs_and_dislikes_cats          24
dislikes_dogs_and_dislikes_cats     17
dislikes_dogs_and_likes_cats         9
dislikes_cats                        5
dislikes_dogs                        5
dislikes_dogs_and_has_cats           4
Name: pets, dtype: int64
```

3.1.4 'language'

In [28]:

```
#language = list(train_df['language'].values)
def lang(language):
    lang_list = []
    for i in language:
        temp = ""
        for j in i.split(','):
            j = j.replace(' ','')
            temp += j.strip() + " "
            temp = temp.replace('(','_')
            temp = temp.replace(')','_')
        lang_list.append(temp.strip())
    return lang_list
```

In [29]:

```
df['cleaned_language'] = lang(list(df['language'].values))
df.drop(['language'], axis=1, inplace=True)
df.head(2)
```

Out[29]:

	user_id	username	age	status	sex	orientation	drinks	drugs	height	job	...	smokes	new_languages	body_profile
0	fffe3100	Edith Lopez	27	single	f	gay	socially	never	66.0	medicine_health	...	no	interested	athletic
1	fffe3200	Travis Young	26	single	m	gay	socially	never	68.0	other	...	no	interested	fit

2 rows × 22 columns

3.1.5 'new_languages'

In [30]:

```
df['new_languages'] = df['new_languages'].str.replace(' ', '_')
df['new_languages'].value_counts()
```

Out[30]:

```
not_interested         1050
interested              633
somewhat_interested     318
Name: new_languages, dtype: int64
```

3.1.6 'body_profile'

In [31]:

```
df['body_profile'] = df['body_profile'].str.replace(' ', '_')
df['body_profile'].value_counts()
```

Out[31]:

```
average           557
fit               433
athletic          385
curvy             179
thin              161
a_little_extra    110
skinny             65
full_figured       43
overweight         25
used_up            20
jacked             13
rather_not_say     10
Name: body_profile, dtype: int64
```

3.1.7 'bio'

In [34]:

```
df['bio']
```

Out[34]:

```
0       bottom line i love life! i work hard and i lov...
1       i'm a straightforward, genuine, fun loving (i'...
2       mmmmm yummy tacosss. yoga is where it's at. i ...
3       i'm a stealth geek. that special mix of techni...
4       with the whisper of the wind i was weaved into...
                              ...
1996    i grew up playing instruments and singing and ...
1997    im a 23 yr old female born and raised in color...
1998    i've spent the last 10 years working in the ou...
1999    i am that i am.  more and more, my spiritual l...
2000    love travel, love food, love wine and love sat...
Name: bio, Length: 2001, dtype: object
```

In [35]:

```
# https://stackoverflow.com/a/47091490/4084039

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [36]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [37]:

```
#printing a few random bios
print(10, df['bio'].values[10])
print('-'*50)
print(1033, df['bio'].values[1033])
print('-'*50)
print(44, df['bio'].values[44])
print('-'*50)
print(777, df['bio'].values[777])
```

```
10 i am a easy-going fun loving, compassionate person. my interests include reading, movies, live
music and game night parties and potlucks. i hope to travel to europe and explore this wonderful w
orld
--------------------------------------------------
1033 in moscow, i had my red october, now it's on to my summer of love! (major points if you know
what that is.) just arrived in oakland (june 1), had to escape phoenix. it was already dante's inf
erno (108f!) and i'm allergic to melting flesh. so i headed for the beautiful bay area to see what
journey was singing about. fun & feisty globetrotting girl lived in italy/europe 6 years
professional dance and acting background i was in a country music video 4ever ago and recently fou
nd it online. i also still get facebook notices from people saying they saw me in "waiting to exha
le". good4laffs!  former royal nanny (really)  int'l english teacher  worked in siberia (not in
the gulag!) and moscow. russian philharmonic dance company guest teacher  taught university
students in south korea (yes, they really do eat dawg! makes the fried silkworms seem blase'.)
ground zero volunteer/fdny supporter  hiked through the rainforest to a waterfall with a jungle bo
y guide (goofy as disneyland & full of useful info when not high on crack)  stayed in a convent wi
th the sisters of mother teresa (weird story, so weird)  stalked by elvis in cyprus/ run-in with d
irty police & my mobster employer/rescued by u.s embassy. i was playing tina turner. sigh. fyi: al
most as many elvises as in vegas.  i have a "cougar" tattoo on my forehead which is only visible t
o men under 30. at least, i figure that must be it. (not that i'm complaining. you young guys are
adorable! equal opportunity romance, here ;-) i hula-hoop by moonlight with palm trees towering ov
er me.
--------------------------------------------------
44 smart sensual sensitive soul, a patron of the arts and an artist, world traveler, financial ana
lyst and a lover of music and dance
--------------------------------------------------
777 i'm a friendly introvert who enjoys people, and i'm a homebody who loves a good adventure. i'm
a tempered optimist, consider myself very fortunate, and i've made it through some pretty tough ti
mes in life with my sense of humor intact.
```

In [38]:

```
def preprocess_text(text_data):
    preprocessed_text = []
    for sentence in tqdm(text_data):
        sent = decontracted(sentence)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in nltk.word_tokenize(sent) if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text
```

In [39]:

```
#nltk.download('punkt')
preprocessed_bio = preprocess_text(df['bio'].values)
```

```
100%|██████████| 2001/2001 [00:01<00:00, 1488.46it/s]
```

In [40]:

```
#printing a few random bios
print(10, preprocessed_bio[10])
print('-'*50)
print(1033, preprocessed_bio[1033])
print('-'*50)
print(44, preprocessed_bio[44])
print('-'*50)
print(777, preprocessed_bio[777])
```

```
10 easy going fun loving compassionate person interests include reading movies live music game nig
ht parties potlucks hope travel europe explore wonderful world
--------------------------------------------------
1033 moscow red october summer love major points know arrived oakland june 1 escape phoenix alread
y dante inferno 108f allergic melting flesh headed beautiful bay area see journey singing fun feis
ty globetrotting girl lived italy europe 6 years professional dance acting background country
music video 4ever ago recently found online also still get facebook notices people saying saw wait
ing exhale good4laffs former royal nanny really int l english teacher worked siberia not gulag mos
cow russian philharmonic dance company guest teacher taught university students south korea yes
really eat dawg makes fried silkworms seem blase ground zero volunteer fdny supporter hiked
rainforest waterfall jungle boy guide goofy disneyland full useful info not high crack stayed conv
ent sisters mother teresa weird story weird stalked elvis cyprus run dirty police mobster employer
rescued u embassy playing tina turner sigh fyi almost many elvises vegas cougar tattoo forehead vi
sible men 30 least figure must not complaining young guys adorable equal opportunity romance hula
hoop moonlight palm trees towering
--------------------------------------------------
44 smart sensual sensitive soul patron arts artist world traveler financial analyst lover music da
nce
--------------------------------------------------
777 friendly introvert enjoys people homebody loves good adventure tempered optimist consider fort
unate made pretty tough times life sense humor intact
```

In [41]:

```
df['bio'] = preprocessed_bio
```

In [ ]:

```
```

3.1.8 'interests'

In [42]:

```
df['interests'] = df['interests'].str.replace(' ', '_')
df['interests'].value_counts()
```

Out[42]:

```
music                155
politics             105
reading              105
singing              101
photography           92
instruments           88
video_games           87
sports                83
painting              83
movies                79
dancing               75
travelling            74
cooking               67
designing             64
craft                 64
sketching             64
food                  60
diy                   49
gardening             48
writting              47
studying              47
yoga                  46
social_networking     46
organising_events     43
makeup                42
acting                41
astronomy             37
fishing               33
calligraphy           33
collectibles          27
camping               16
Name: interests, dtype: int64
```

3.1.9 'other_interests'

In [43]:

```
df['other_interests']=df['other_interests'].str.replace(' ','_')
df['other_interests'].value_counts()
```

Out[43]:

```
music                158
sports               102
reading              100
photography          100
singing               98
politics              94
video_games           90
dancing               88
painting              85
movies                81
craft                 78
instruments           72
sketching             67
travelling            62
designing             61
makeup                57
social_networking     57
food                  54
diy                   53
writting              50
gardening             49
cooking               49
acting                45
fishing               38
studying              36
yoga                  35
astronomy             35
organising_events     34
calligraphy           33
collectibles          21
```

```
camping                19
Name: other_interests, dtype: int64
```

3.1.10 'location_preference'

In [44]:

```
df['location_preference'] = df['location_preference'].str.replace(' ', '_')
df['location_preference'].value_counts()
```

Out[44]:

```
same_city     1169
same_state     495
anywhere       337
Name: location_preference, dtype: int64
```

3.1.11 'status'

In [45]:

```
df['status'] = df['status'].str.replace(' ', '_')
df['status'].value_counts()
```

Out[45]:

```
single            1867
available           66
seeing_someone      54
married             14
Name: status, dtype: int64
```

3.2 Preprocessing Numerical Data

3.2.1 'age'

In [46]:

```
scaler = StandardScaler()
scaler.fit(df['age'].values.reshape(-1, 1))
df['age']=scaler.transform(df['age'].values.reshape(-1, 1))
```

In [47]:

```
df['age'].head(10)
```

Out[47]:

```
0   -0.579402
1   -0.674817
2   -1.247305
3   -0.579402
4   -1.056475
5   -1.342719
6    0.756403
7    1.042647
8   -0.865646
9   -0.483987
Name: age, dtype: float64
```

3.2.2 'height'

In [48]:

```
scaler = StandardScaler()
scaler.fit(df['height'].values.reshape(-1, 1))
df['height']=scaler.transform(df['height'].values.reshape(-1, 1))
```

In [49]:

```
df['height'].head(10)
```

Out[49]:

```
0   -0.588006
1   -0.074431
2    0.182356
3   -0.074431
4   -0.074431
5   -0.588006
6   -1.871943
7    0.182356
8   -0.074431
9   -1.615155
Name: height, dtype: float64
```

3.2.3 'education_level'

In [50]:

```
scaler = StandardScaler()
scaler.fit(df['education_level'].values.reshape(-1, 1))
df['education_level']=scaler.transform(df['education_level'].values.reshape(-1, 1))
```

In [51]:

```
df['education_level'].head(10)
```

Out[51]:

```
0    1.314342
1    0.083069
2   -1.148204
3    0.083069
4   -1.148204
5   -1.148204
6    0.083069
7    0.083069
8    0.083069
9    1.314342
Name: education_level, dtype: float64
```

In [52]:

```
df.head(1)
```

Out[52]:

	user_id	username	age	status	sex	orientation	drinks	drugs	height	job	...	smokes	new_languages	body_
0	fffe3100	Edith Lopez	-0.579402	single	f	gay	socially	never	-0.588006	medicine_health	...	no	interested	

1 rows × 22 columns

In [ ]:

```
```

In [7]:

```
if not os.path.isfile('train.csv'):
    # create the dataframe and store it in the disk for offline purposes..
    df.to_csv("train.csv", index=False)
```

In [8]:

```
df = pd.read_csv("train.csv")
```

In [9]:

```
df.shape
```

Out[9]:

```
(2001, 22)
```

4. Vectorizing Categorial features

4.1 Doc2Vec on 'bio'

In [79]:

```

```

In [10]:

```
df.loc[1558]
```

Out[10]:

```
user_id                fffe3100350035003900
username                         Sandra Lowe
age                                  1.71055
status                                single
sex                                        f
orientation                         straight
drinks                              socially
drugs                                  never
height                             -1.10158
job                       education_academia
location                berkeley  california
pets                                has_cats
smokes                                    no
new_languages                not_interested
body_profile                        average
education_level                   0.0830694
dropped_out                              no
bio                                     NaN
interests                          designing
other_interests                  video_games
location_preference                 anywhere
cleaned_language          english_fluently_
Name: 1558, dtype: object
```

In [11]:

```
df["bio"]=df['bio'].replace(np.nan, "0")
```

In [12]:

```
w2v_total_data = list(df['bio'])
```

In [13]:

```
def build_model(max_epochs, vec_size, alpha, tagged_data):

    model = Doc2Vec(vector_size=vec_size,
                alpha=alpha,
                min_alpha=0.00025,
                min_count=2,
                dm=1)

    model.build_vocab(tag_data)

    # With the model built we simply train on the data.

    for epoch in range(max_epochs):
        print(f"Iteration {epoch}")
        model.train(tag_data,
                    total_examples=model.corpus_count,
                    epochs=model.epochs)

        # Here I decrease the learning rate.

        model.alpha -= 0.0002

        model.min_alpha = model.alpha

    # Now simply save the model to avoid training again.

    model.save("d2v.model")
    print("Model Saved")
    return model
```

In [14]:

```
tag_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(w
2v_total_data)]
```

In [15]:

```
model = build_model(max_epochs=5, vec_size=50, alpha=0.025, tagged_data=tag_data)
```

```
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Model Saved
```

In [16]:

```
d2v = Doc2Vec.load("d2v.model")
```

In [17]:

```
#vectorising the df['bio']
bio_vectorized = [d2v.docvecs[str(i)] for i in range(len(df['bio']))]
```

In [18]:

```
np.asarray(bio_vectorized).shape
```

Out[18]:

```
(2001, 50)
```

4.2 Word2Vec on 'location'

In [13]:

```
tokenized_locations = [word_tokenize(df['location'][i]) for i in range(len(df['location']))]
```

In [14]:

```
def w2v_locations(location_i_tokenized):
    vectorized_location_i = 0
    for word in location_i_tokenized:
        try:
            vectorized_location_i += w2v[word]
        except:
            w2v[word] = np.zeros_like(w2v['word'])
        finally:
            vectorized_location_i += w2v[word]
    return vectorized_location_i
```

In [17]:

```
vectorized_locations = [w2v_locations(location_i_tokenized) for location_i_tokenized in
tokenized_locations]
```

In [76]:

```
with open('vectorized_locs.pickle', 'wb') as f:
    pickle.dump(vectorized_locations, f)
```

In [19]:

```
with open('vectorized_locs.pickle' ,"rb", buffering=0) as f:
    vectorized_locations = pickle.load(f)
```

In [54]:

```
df.head(3)
```

Out[54]:

	user_id	username	age	status	sex	orientation	drinks	drugs	height	job	...	smokes	new_la
0	fffe3100	Edith Lopez	-0.579402	single	f	gay	socially	never	-0.588006	medicine_health	...	no	i
1	fffe3200	Travis Young	-0.674817	single	m	gay	socially	never	-0.074431	other	...	no	i
2	fffe3300	Agnes Smith	-1.247305	seeing_someone	f	bisexual	socially	sometimes	0.182356	other	...	sometimes	i

3 rows × 22 columns

In [21]:

```
df.columns
```

Out[21]:

```
Index(['user_id', 'username', 'age', 'status', 'sex', 'orientation', 'drinks',
       'drugs', 'height', 'job', 'location', 'pets', 'smokes', 'new_languages',
       'body_profile', 'education_level', 'dropped_out', 'bio', 'interests',
       'other_interests', 'location_preference', 'cleaned_language'],
      dtype='object')
```

4.3 Word2Vec on 'cleaned_language'

In [82]:

```
df['cleaned_language'] = df['cleaned_language'].str.replace('_', ' ')
```

In [83]:

```
tokenized_langs = [word_tokenize(df['cleaned_language'][i]) for i in
range(len(df['cleaned_language']))]
```

In [84]:

```
vectorized_language = [w2v_locations(location_i_tokenized) for location_i_tokenized in
tokenized_langs]
```

In [85]:

```
with open('vectorized_langs.pickle', 'wb') as f:
    pickle.dump(vectorized_language, f)
```

In [20]:

```
with open('vectorized_langs.pickle' ,"rb", buffering=0) as f:
    vectorized_language = pickle.load(f)
```

In [ ]:

```

```

4.4 One hot encoding the rest of the features

In [21]:

```
from sklearn.preprocessing import OneHotEncoder
cat_features = ['status','drinks','drugs','job','pets','smokes','new_languages','body_profile','dro
pped_out','interests','other_interests','location_preference']
encoded_features = []


for feature in cat_features:
        encoded_feat = OneHotEncoder().fit_transform(df[feature].values.reshape(-1, 1)).toarray()
        n = df[feature].nunique()
        cols = ['{}_{}'.format(feature, n) for n in range(1, n + 1)]
        encoded_df = pd.DataFrame(encoded_feat, columns=cols)
        encoded_df.index = df.index
        encoded_features.append(encoded_df)
df1= pd.concat([df, *encoded_features[:12]], axis=1)
```

In [22]:

```
df1.drop(['status','drinks','drugs','job','pets','smokes','new_languages','body_profile','dropped_o
ut','interests','other_interests','location_preference'],axis=1,inplace=True)
```

In [23]:

```
cols = ['{}_{}'.format("bio_word", n) for n in range(1, 50 + 1)]
encoded_df = pd.DataFrame(bio_vectorized, columns=cols)
encoded_df.index = df1.index
```

In [24]:

```
df1=pd.concat((df1,encoded_df), axis=1)
```

In [25]:

```
cols = ['{}_{}'.format("lang", n) for n in range(1, 300 + 1)]
```

```
encoded_df = pd.DataFrame(vectorized_language, columns=cols)
encoded_df.index = df1.index
```

In [26]:

```
df1=pd.concat((df1,encoded_df), axis=1)
```

In [27]:

```
cols = ['{}_{}'.format("loc", n) for n in range(1, 300 + 1)]
encoded_df = pd.DataFrame(vectorized_locations, columns=cols)
encoded_df.index = df1.index
```

In [28]:

```
df1=pd.concat((df1,encoded_df), axis=1)
```

In [29]:

```
df1.drop(["location"],axis=1,inplace=True)
```

In [30]:

```
df1.drop(["bio","cleaned_language"],axis=1,inplace=True)
```

In [31]:

```
df1.drop(["username"],axis=1,inplace=True)
```

In [48]:

```
temp=df1[["user_id","sex","orientation"]]
```

In [50]:

```
df1.drop(["sex","orientation","user_id"],axis=1,inplace=True)
```

In [68]:

```
if (temp.loc[0,"sex"]=='f') and (temp.loc[0,"orientation"]=="gay"):
    print(cosine_similarity([df1.loc[0]],[df1.loc[1]])[0][0])
else:
    print("0")
```

```
0.8215193398048564
```

Cosine similarity logic

In [42]:

```
from sklearn.metrics.pairwise import cosine_similarity
```

In [69]:

```
ans=[]
for i in range(0,len(temp)):
    l=[]
    for j in range(0,len(temp)):
        if (temp.loc[i,"orientation"]=="gay") and (temp.loc[j,"orientation"]=="gay"):
            if ((temp.loc[i,"sex"]=='m') and (temp.loc[j,"sex"]=='f'))or((temp.loc[i,"sex"]=='f') a
nd (temp.loc[j,"sex"]=='m')):
                l.append(0.0)
            else:
                l.append(cosine_similarity([df1.loc[i]],[df1.loc[j]])[0][0])
```

```
        elif (temp.loc[i,"orientation"]=="straight") and (temp.loc[j,"orientation"]=="straight"):
            if ((temp.loc[i,"sex"]=='m') and (temp.loc[j,"sex"]=='m'))or((temp.loc[i,"sex"]=='f') a
nd (temp.loc[j,"sex"]=='f')):
                l.append(0.0)
            else:
                l.append(cosine_similarity([df1.loc[i]],[df1.loc[j]])[0][0])
        elif ((temp.loc[i,"orientation"]=="gay") and (temp.loc[j,"orientation"]=="bisexual")) or ((
temp.loc[i,"orientation"]=="bisexual") and (temp.loc[j,"orientation"]=="gay")):
            if ((temp.loc[i,"sex"]=='m') and (temp.loc[j,"sex"]=='f'))or((temp.loc[i,"sex"]=='f') a
nd (temp.loc[j,"sex"]=='m')):
                l.append(0.0)
            else:
                l.append(cosine_similarity([df1.loc[i]],[df1.loc[j]])[0][0])
        elif ((temp.loc[i,"orientation"]=="straight") and (temp.loc[j,"orientation"]=="bisexual"))
or ((temp.loc[i,"orientation"]=="bisexual") and (temp.loc[j,"orientation"]=="straight")):
            if ((temp.loc[i,"sex"]=='m') and (temp.loc[j,"sex"]=='m'))or((temp.loc[i,"sex"]=='f') a
nd (temp.loc[j,"sex"]=='f')):
                l.append(0.0)
            else:
                l.append(cosine_similarity([df1.loc[i]],[df1.loc[j]])[0][0])
        elif ((temp.loc[i,"orientation"]=="gay") and (temp.loc[j,"orientation"]=="straight")) or ((
temp.loc[i,"orientation"]=="straight") and (temp.loc[j,"orientation"]=="gay")):
            l.append(0.0)
        else:
            l.append(cosine_similarity([df1.loc[i]],[df1.loc[j]])[0][0])
    ans.append(l)
```

In [86]:

```
ans1=np.array(ans)
```

In [104]:

```
result=ans1
```

In [105]:

```
column_values = list(temp['user_id'])
df_to_submit = pd.DataFrame(data = result,
                  columns = column_values)
```

In [106]:

```
df_to_submit
```

Out[106]:

	fffe3100	fffe3200	fffe3300	fffe3400	fffe3500	fffe3600	fffe3700	fffe3800	fffe3900	fffe31003000	...	fffe3100390039003200
0	1.000000	0.000000	0.919758	0.000000	0.689420	0.000000	0.808239	0.000000	0.000000	0.000000	...	0.000000
1	0.000000	1.000000	0.000000	0.610969	0.000000	0.752582	0.000000	0.000000	0.800954	0.000000	...	0.780685
2	0.919758	0.000000	1.000000	0.543779	0.592012	0.700426	0.686190	0.698014	0.000000	0.673389	...	0.000000
3	0.000000	0.610969	0.543779	1.000000	0.838891	0.754554	0.000000	0.000000	0.726056	0.000000	...	0.652668
4	0.689420	0.000000	0.592012	0.838891	1.000000	0.781003	0.863628	0.871552	0.000000	0.687035	...	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...
1996	0.000000	0.000000	0.578207	0.000000	0.956082	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
1997	0.615925	0.000000	0.513020	0.000000	0.821472	0.000000	0.746619	0.000000	0.000000	0.000000	...	0.000000
1998	0.000000	0.000000	0.475716	0.000000	0.714520	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
1999	0.000000	0.000000	0.698018	0.000000	0.723237	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
2000	0.000000	0.000000	0.000000	0.723345	0.000000	0.868945	0.000000	0.896126	0.000000	0.741266	...	0.000000

2001 rows × 2001 columns

making diagonal values to 0 since similairty value with themselves may be 100% but we dont recommend a person to themselves in recommendation system

In [107]:

```
df_to_submit.values[[np.arange(df_to_submit.shape[0])]*2] = 0.0
```

In [108]:

```
df_to_submit.insert(0,"user_id",temp["user_id"].values)
```

In [109]:

```
df_to_submit
```

Out[109]:

	user_id	fffe3100	fffe3200	fffe3300	fffe3400	fffe3500	fffe3600	fffe3700	fffe3800	fffe3900	...	fffe31003900
0	fffe3100	0.000000	0.000000	0.919758	0.000000	0.689420	0.000000	0.808239	0.000000	0.000000	...	
1	fffe3200	0.000000	0.000000	0.000000	0.610969	0.000000	0.752582	0.000000	0.000000	0.800954	...	
2	fffe3300	0.919758	0.000000	0.000000	0.543779	0.592012	0.700426	0.686190	0.698014	0.000000	...	
3	fffe3400	0.000000	0.610969	0.543779	0.000000	0.838891	0.754554	0.000000	0.000000	0.726056	...	
4	fffe3500	0.689420	0.000000	0.592012	0.838891	0.000000	0.781003	0.863628	0.871552	0.000000	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1996	fffe3100390039003700	0.000000	0.000000	0.578207	0.000000	0.956082	0.000000	0.000000	0.000000	0.000000	...	
1997	fffe3100390039003800	0.615925	0.000000	0.513020	0.000000	0.821472	0.000000	0.746619	0.000000	0.000000	...	
1998	fffe3100390039003900	0.000000	0.000000	0.475716	0.000000	0.714520	0.000000	0.000000	0.000000	0.000000	...	
1999	fffe3200300030003000	0.000000	0.000000	0.698018	0.000000	0.723237	0.000000	0.000000	0.000000	0.000000	...	
2000	fffe3200300030003100	0.000000	0.000000	0.000000	0.723345	0.000000	0.868945	0.000000	0.896126	0.000000	...	

2001 rows × 2002 columns

In [97]:

```
df_to_submit.to_csv('submission.csv')
```

Conclusion

In [1]:

```
from prettytable import PrettyTable
```

In [2]:

```
pt=PrettyTable()
pt.field_names=["score=max(0,100-RMSE(actual,predicted)"]
pt.add_row(["97.86"])
print(pt)
```

```
+----------------------------------------+
| score=max(0,100-RMSE(actual,predicted) |
+----------------------------------------+
|                 97.86                  |
+----------------------------------------+
```

In [ ]:

```

```