nigwd4vvt

March 28, 2023

[2]:
```
import numpy as np
import pandas as pd
```

[3]:
```
df=pd.read_csv('C:/Users/KIIT/OneDrive/Desktop/MY STUDIES/TalentBattle/Week 3/
 ↪stress.csv')
df.head()
```

[3]:

	subreddit	post_id	sentence_range	text	id	label	confidence	social_timestamp	social_karma	syntax_ari	…	lex_dal_min_pleasantness	lex_dal_min_activation	lex_dal_min_imagery	lex_dal_avg_activation	lex_dal_avg_imagery	lex_dal_avg_pleasantness
0	ptsd	8601tu	(15, 20)	He said he had not felt that way before, sugge…	33181	1	0.8	1521614353	5	1.806818	…	1.000	1.1250	1.0	1.77000	1.52211	1.89556
1	assistance	8lbrx9	(0, 5)	Hey there r/assistance, Not sure if this is th…	2606	0	1.0	1527009817	4	9.429737	…	1.125	1.0000	1.0			
2	ptsd	9ch1zh	(15, 20)	My mom then hit me with the newspaper and it s…	38816	1	0.8	1535935605	2	7.769821	…	1.000	1.1429	1.0			
3	relationships	7rorpp	[5, 10]	until i met my new boyfriend, he is amazing, h…	239	1	0.6	1516429555	0	2.667798	…	1.000	1.1250	1.0			
4	survivorsofabuse	9p2gbc	[0, 5]	October is Domestic Violence Awareness Month a…	1421	1	0.8	1539809005	24	7.554238	…	1.000	1.1250	1.0			

```
1                 1.69586              1.62045                   1.88919
2                 1.83088              1.58108                   1.85828
3                 1.75356              1.52114                   1.98848
4                 1.77644              1.64872                   1.81456

   social_upvote_ratio  social_num_comments  syntax_fk_grade  sentiment
0                 0.86                    1         3.253573  -0.002742
1                 0.65                    2         8.828316   0.292857
2                 0.67                    0         7.841667   0.011894
3                 0.50                    5         4.104027   0.141671
4                 1.00                    1         7.910952  -0.204167

[5 rows x 116 columns]
```

[4]:
```
df.describe()
```

[4]:
```
                id        label   confidence  social_timestamp  social_karma  \
count   2838.000000  2838.000000  2838.000000      2.838000e+03   2838.000000
mean   13751.999295     0.524313     0.808972      1.518107e+09     18.262156
std    17340.161897     0.499497     0.177038      1.552209e+07     79.419166
min        4.000000     0.000000     0.428571      1.483274e+09      0.000000
25%      926.250000     0.000000     0.600000      1.509698e+09      2.000000
50%     1891.500000     1.000000     0.800000      1.517066e+09      5.000000
75%    25473.750000     1.000000     1.000000      1.530898e+09     10.000000
max    55757.000000     1.000000     1.000000      1.542592e+09   1435.000000

        syntax_ari  lex_liwc_WC  lex_liwc_Analytic  lex_liwc_Clout  \
count  2838.000000  2838.000000        2838.000000     2838.000000
mean      4.684272    85.996124          35.240941       40.948231
std       3.316435    32.334887          26.486189       31.587117
min      -6.620000     5.000000           1.000000        1.000000
25%       2.464243    65.000000          12.410000       12.135000
50%       4.321886    81.000000          29.420000       33.520000
75%       6.505657   101.000000          55.057500       69.320000
max      24.074231   310.000000          99.000000       99.000000

       lex_liwc_Authentic  …  lex_dal_min_pleasantness  \
count         2838.000000  …               2838.000000
mean            67.044249  …                  1.088001
std             32.880644  …                  0.117159
min              1.000000  …                  1.000000
25%             41.070000  …                  1.000000
50%             80.710000  …                  1.000000
75%             96.180000  …                  1.142900
max             99.000000  …                  1.900000

       lex_dal_min_activation  lex_dal_min_imagery  lex_dal_avg_activation  \
```

```
count             2838.000000          2838.000000             2838.000000
mean                 1.120099             1.000211                1.722759
std                  0.085227             0.006500                0.047835
min                  1.000000             1.000000                1.485400
25%                  1.000000             1.000000                1.691430
50%                  1.142900             1.000000                1.721430
75%                  1.142900             1.000000                1.751760
max                  1.500000             1.200000                2.007400

       lex_dal_avg_imagery  lex_dal_avg_pleasantness  social_upvote_ratio  \
count          2838.000000               2838.000000          2838.000000
mean              1.536400                  1.879385             0.843517
std               0.102971                  0.058932             0.174794
min               1.200000                  1.561150             0.140000
25%               1.469745                  1.841782             0.750000
50%               1.530295                  1.878250             0.890000
75%               1.596030                  1.916243             1.000000
max               2.066670                  2.158490             1.000000

       social_num_comments  syntax_fk_grade    sentiment
count          2838.000000      2838.000000  2838.000000
mean              9.948555         5.448836     0.040740
std              21.798032         2.535829     0.195490
min               0.000000        -1.918000    -1.000000
25%               2.000000         3.729973    -0.072222
50%               5.000000         5.210000     0.044821
75%              10.000000         6.855217     0.166667
max             416.000000        21.198919     1.000000

[8 rows x 112 columns]
```

[5]:
```
df.isnull().sum()
```

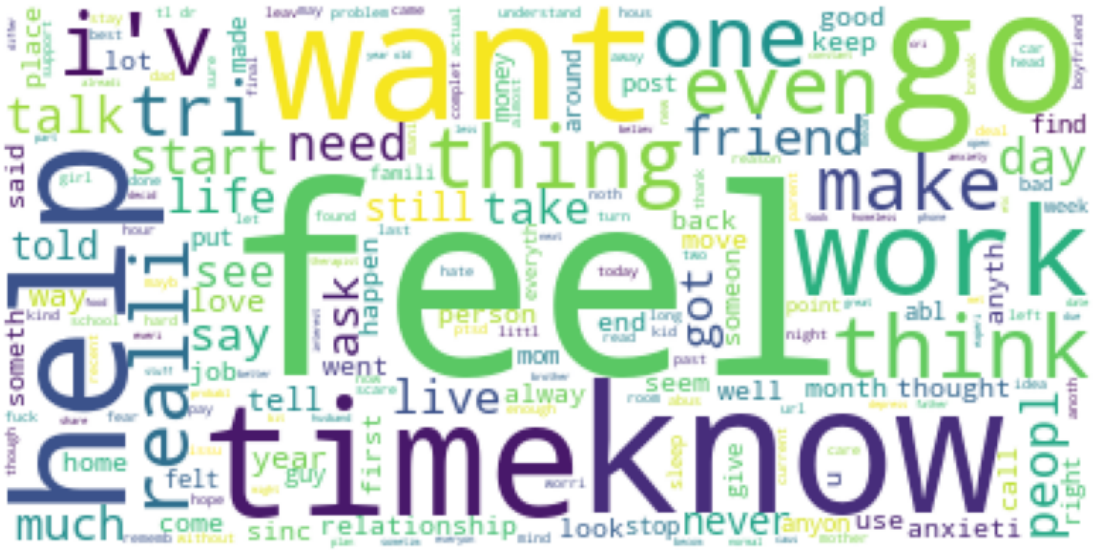
[5]:
```
subreddit                   0
post_id                     0
sentence_range              0
text                        0
id                          0
                           ..
lex_dal_avg_pleasantness    0
social_upvote_ratio         0
social_num_comments         0
syntax_fk_grade             0
sentiment                   0
Length: 116, dtype: int64
```

[6]:
```
import nltk
import re
from nltk. corpus import stopwords
import string
nltk. download( 'stopwords' )
stemmer = nltk. SnowballStemmer("english")
stopword=set (stopwords . words ( 'english' ))

def clean(text):
    text = str(text) . lower()  #returns a string where all characters are␣
 ↪lower case. Symbols and Numbers are ignored.
    text = re. sub('\[.*?\]',' ',text)  #substring and returns a string with␣
 ↪replaced values.
    text = re. sub('https?://\S+/www\. \S+', ' ', text)#whitespace char with␣
 ↪pattern
    text = re. sub('<. *?>+', ' ', text)#special char enclosed in square␣
 ↪brackets
    text = re. sub(' [%s]' % re. escape(string. punctuation), ' ',␣
 ↪text)#eliminate punctuation from string
    text = re. sub(' \n',' ', text)
    text = re. sub(' \w*\d\w*' ,' ', text)#word character ASCII punctuation
    text = [word for word in text. split(' ') if word not in stopword] ␣
 ↪#removing stopwords
    text =" ". join(text)
    text = [stemmer . stem(word) for word in text. split(' ') ]#remove␣
 ↪morphological affixes from words
    text = " ". join(text)
    return text
df [ "text"] = df["text"]. apply(clean)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\KIIT\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[8]:
```
import matplotlib. pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
text = " ". join(i for i in df. text)
stopwords = set (STOPWORDS)
wordcloud = WordCloud( stopwords=stopwords,background_color="white") .␣
 ↪generate(text)
plt. figure(figsize=(10, 10) )
plt. imshow(wordcloud )
plt. axis("off")
plt. show( )
```



[9]:
```
from sklearn. feature_extraction. text import CountVectorizer
from sklearn. model_selection import train_test_split

x = np.array (df["text"])
y = np.array (df["label"])

cv = CountVectorizer ()
X = cv. fit_transform(x)
print(X)
xtrain, xtest, ytrain, ytest = train_test_split(X, y,test_size=0.33)
```

```
  (0, 7405)     1
  (0, 3278)     1
  (0, 9454)     1
  (0, 861)      1
  (0, 8359)     1
  (0, 3750)     1
  (0, 7214)     1
  (0, 8908)     1
  (0, 298)      1
  (0, 9749)     1
  (0, 4303)     1
  (0, 5034)     1
  (0, 5325)     1
  (0, 2188)     1
  (0, 5118)     1
  (0, 3265)     1
  (0, 2593)     3
```

```
  (0, 4188)     1
  (0, 5316)     1
  (0, 3697)     1
  (0, 8339)     1
  (0, 6861)     1
  (0, 4150)     1
  (0, 5174)     1
  (0, 1831)     1
  :     :
  (2836, 877)   1
  (2836, 4555)  1
  (2836, 2928)  1
  (2836, 4615)  1
  (2836, 4785)  1
  (2836, 4511)  1
  (2837, 7405)  2
  (2837, 3018)  1
  (2837, 5533)  2
  (2837, 8784)  1
  (2837, 8502)  1
  (2837, 6770)  1
  (2837, 4318)  1
  (2837, 9670)  1
  (2837, 5569)  1
  (2837, 8881)  1
  (2837, 5713)  1
  (2837, 2587)  1
  (2837, 7468)  1
  (2837, 2351)  1
  (2837, 7804)  1
  (2837, 2758)  1
  (2837, 8880)  1
  (2837, 5459)  1
  (2837, 3020)  1
```

[10]:
```
from sklearn.naive_bayes import BernoulliNB
model=BernoulliNB()
model.fit(xtrain,ytrain)
```

[10]:
```
BernoulliNB()
```

[11]:
```
user=input("Enter the text : ")
data=cv.transform([user]).toarray()
output=model.predict(data)
print(output)
```

```
Enter the text : sad
[1]
```

[12]:
```
user=input("Enter the text : ")
data=cv.transform([user]).toarray()
output=model.predict(data)
print(output)
```

```
Enter the text : i think we need to take care of ourselves
[0]
```

[13]:
```
user=input("Enter the text : ")
data=cv.transform([user]).toarray()
output=model.predict(data)
print(output)
```

```
Enter the text : some times i feel like i need help
[1]
```