Stress Detection with Machine Learning

Sreeparna Ray

Stress is a prevalent issue that can have detrimental effects on an individual's health, productivity, and overall quality of life. However, detecting and managing stress is often challenging because it is a subjective experience and can manifest differently in each person. Traditional methods of stress assessment rely on self-reporting, which can be unreliable and prone to biases.

Machine learning-based stress detection offers several advantages over conventional approaches. It provides an objective and data-driven analysis of stress levels, bypassing the limitations of self-reporting. By automating the stress detection process, individuals can receive timely interventions, leading to better stress management and improved well-being. Now I will walk through the task of stress detection with machine learning using python.

Now let's start the task of stress detection with machine learning. I will start this task by importing the necessary Python libraries and the dataset that we need for this task:

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv("stress.csv")
print(data.head())
```

```
          subreddit post_id sentence_range  \
0              ptsd  8601tu       (15, 20)
1       assistance  8lbrx9         (0, 5)
2              ptsd  9ch1zh       (15, 20)
3    relationships  7rorpp        [5, 10]
4  survivorsofabuse  9p2gbc         [0, 5]

                                                text     id  label  \
0  He said he had not felt that way before, sugge...  33181      1
1  Hey there r/assistance, Not sure if this is th...   2606      0
2  My mom then hit me with the newspaper and it s...  38816      1
3  until i met my new boyfriend, he is amazing, h...    239      1
4  October is Domestic Violence Awareness Month a...   1421      1

   confidence  social_timestamp  social_karma  syntax_ari  ...  \
0         0.8        1521614353             5    1.806818  ...
1         1.0        1527009817             4    9.429737  ...
2         0.8        1535935605             2    7.769821  ...
3         0.6        1516429555             0    2.667798  ...
4         0.8        1539809005            24    7.554238  ...

   lex_dal_min_pleasantness  lex_dal_min_activation  lex_dal_min_imagery  \
0                     1.000                  1.1250                  1.0
1                     1.125                  1.0000                  1.0
2                     1.000                  1.1429                  1.0
3                     1.000                  1.1250                  1.0
4                     1.000                  1.1250                  1.0

   lex_dal_avg_activation  lex_dal_avg_imagery  lex_dal_avg_pleasantness  \
0                 1.77000              1.52211                   1.89556
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]
```

In [3]:
```
# Check whether this dataset contains any null values or not:
print(data.isnull().sum())
```

```
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
```

So this dataset does not have any null values.

Let's prepare the text column of this dataset to clean the text column with stopwords, links, special symbols and language errors:

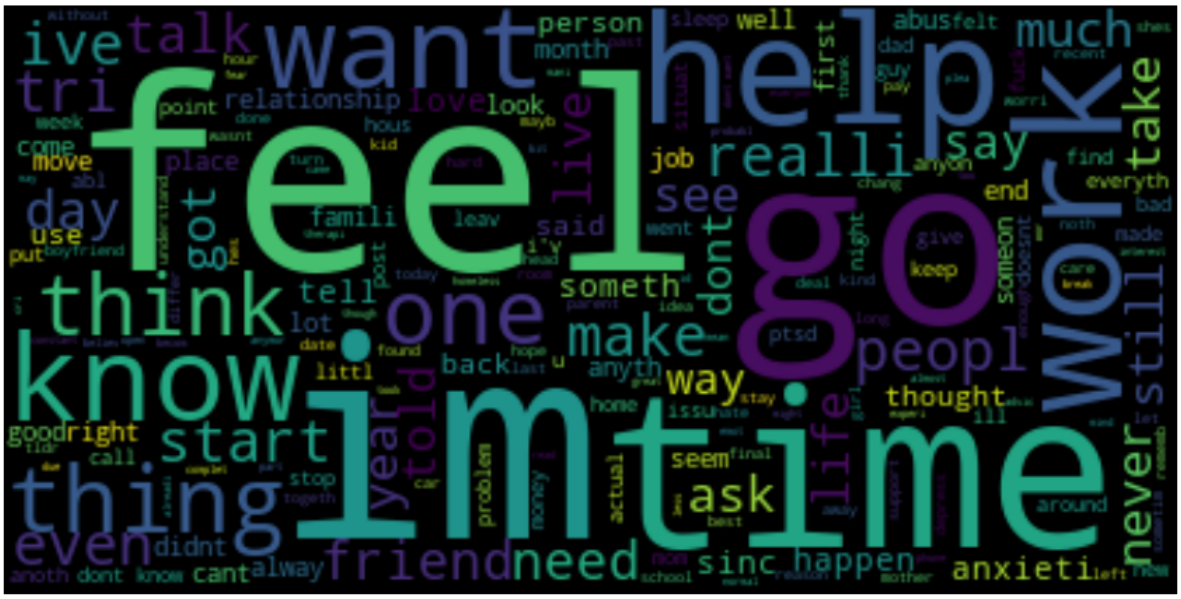
In [4]:
```
import nltk
import re
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))
```

```
[nltk_data] Error loading stopwords: <urlopen error [WinError 10060] A
[nltk_data]     connection attempt failed because the connected party
[nltk_data]     did not properly respond after a period of time, or
[nltk_data]     established connection failed because connected host
[nltk_data]     has failed to respond>
```

In [5]:
```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["text"] = data["text"].apply(clean)
```

Let's have a look at the most used words by the people sharing about their life problems on social media by visualizing a word cloud of the text column:

In [6]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
text = " ".join(i for i in data.text)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,
                      background_color="black").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Stress Detection Model

The label column in this dataset contains labels as 0 and 1. 0 means no stress, and 1 means stress. I will use Stress and No stress labels instead of 1 and 0. So let's prepare this column accordingly and select the text and label columns for the process of training a machine learning model:

In [7]:
```
data["label"] = data["label"].map({0: "No Stress", 1: "Stress"})
data = data[["text", "label"]]
print(data.head())
```

```
                                                text      label
0  said felt way sugget go rest trigger ahead you...     Stress
1  hey rassist sure right place post goe  im curr...  No Stress
2  mom hit newspap shock would know dont like pla...     Stress
3  met new boyfriend amaz kind sweet good student...     Stress
4  octob domest violenc awar month domest violenc...     Stress
```

Now I will split this dataset into training and test sets:

In [8]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

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

cv = CountVectorizer()
X = cv.fit_transform(x)
xtrain, xtest, ytrain, ytest = train_test_split(X, y,
                                                test_size=0.33,
                                                random_state=42)
```

As this task is based on the problem of binary classification, I will be using the Bernoulli Naive Bayes algorithm, which is one of the best algorithms for binary classification problems. So let's train the stress detection model:

In [9]:
```
from sklearn.naive_bayes import BernoulliNB
model = BernoulliNB()
model.fit(xtrain, ytrain)
```

Out[9]:
```
BernoulliNB()
```

Now let's test the performance of our model on some random sentences based on mental health:

In [10]:
```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: I am really busy today
['No Stress']
```

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

```
Enter a Text: Sometimes I feel very uncomfortable like I can't do anything
['Stress']
```

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

```
Enter a Text: Sometimes I feel like I have no one who can stay with me
['Stress']
```

In [ ]:
```

```