STRESS DETECTION MACHINE LEARNING MODEL

Problem statement

Predicting various human stress behaviour patterns

In [1]:
```
# upload the needed libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

```
C:\Users\HP\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and
<1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
# load dataset
df = pd.read_csv(r'C:\Users\HP\Desktop\Data Science\Datasets\Stress.csv')
```

In [3]:
```
df
```

Out[3]:

	subreddit	post_id	sentence_range	text	label	confidence	social_timestamp
0	ptsd	8601tu	(15, 20)	He said he had not felt that way before, sugge...	1	0.800000	1521614353
1	assistance	8lbrx9	(0, 5)	Hey there r/assistance, Not sure if this is th...	0	1.000000	1527009817
2	ptsd	9ch1zh	(15, 20)	My mom then hit me with the newspaper and it s...	1	0.800000	1535935605
3	relationships	7rorpp	[5, 10]	until i met my new boyfriend, he is amazing, h...	1	0.600000	1516429555
4	survivorsofabuse	9p2gbc	[0, 5]	October is Domestic Violence Awareness Month a...	1	0.800000	1539809005
...	...	...	...	...	...	...	...
2833	relationships	7oee1t	[35, 40]	* Her, a week ago: Precious, how are you? (I i...	0	1.000000	1515187044
2834	ptsd	9p4ung	[20, 25]	I don't have the ability to cope with it anymo...	1	1.000000	1539827412
2835	anxiety	9nam6l	(5, 10)	In case this is the first time you're reading ...	0	1.000000	1539269312
2836	almosthomeless	5y53ya	[5, 10]	Do you find this normal? They have a good rela	0	0.571429	1488938143

In [4]:
```
# check general information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2838 entries, 0 to 2837
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   subreddit         2838 non-null   object 
 1   post_id           2838 non-null   object 
 2   sentence_range    2838 non-null   object 
 3   text              2838 non-null   object 
 4   label             2838 non-null   int64  
 5   confidence        2838 non-null   float64
 6   social_timestamp  2838 non-null   int64  
dtypes: float64(1), int64(2), object(4)
memory usage: 155.3+ KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(2838, 7)
```

Exploratory Data Analysis

In [6]:
```
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
label	2838.0	5.243129e-01	4.994965e-01	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
confidence	2838.0	8.089718e-01	1.770383e-01	4.285714e-01	6.000000e-01	8.000000e-01	1.000000e+00	1.000000e+00
social_timestamp	2838.0	1.518107e+09	1.552209e+07	1.483274e+09	1.509698e+09	1.517066e+09	1.530898e+09	1.542592e+09

In [7]:
```
# print columns features
df.columns.tolist()
```

Out[7]:
```
['subreddit',
 'post_id',
 'sentence_range',
 'text',
 'label',
 'confidence',
 'social_timestamp']
```

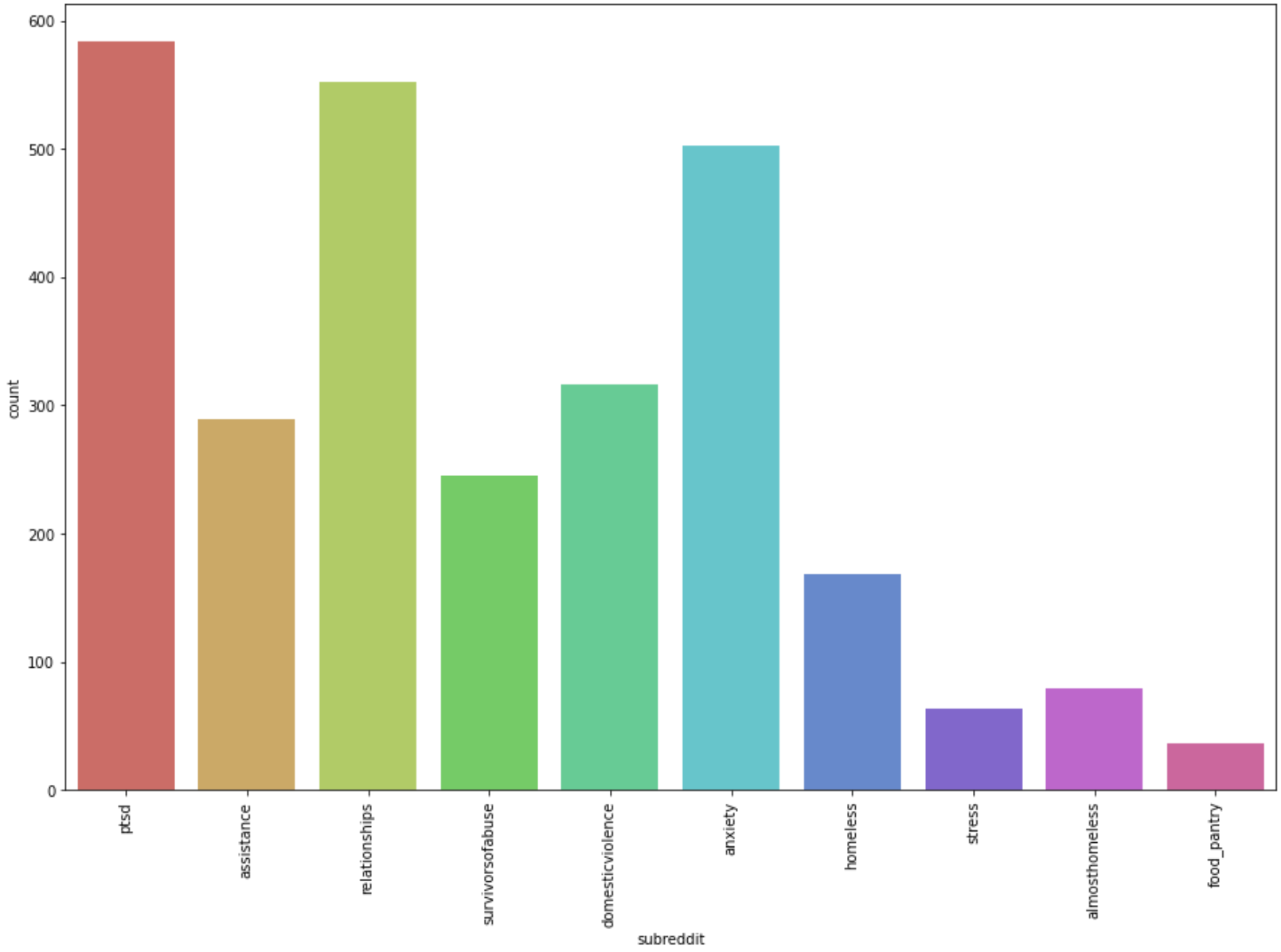
In [8]:
```
# print unique features of subreddit
df['subreddit'].unique().tolist()
```

Out[8]:
```
['ptsd',
 'assistance',
 'relationships',
 'survivorsofabuse',
 'domesticviolence',
 'anxiety',
 'homeless',
 'stress',
 'almosthomeless',
 'food_pantry']
```

In [9]:
```
# check and visualize the count of values
df['subreddit'].value_counts()
```

Out[9]:
```
ptsd                584
relationships       552
anxiety             503
domesticviolence    316
assistance          289
survivorsofabuse    245
homeless            168
almosthomeless       80
stress               64
food_pantry          37
Name: subreddit, dtype: int64
```

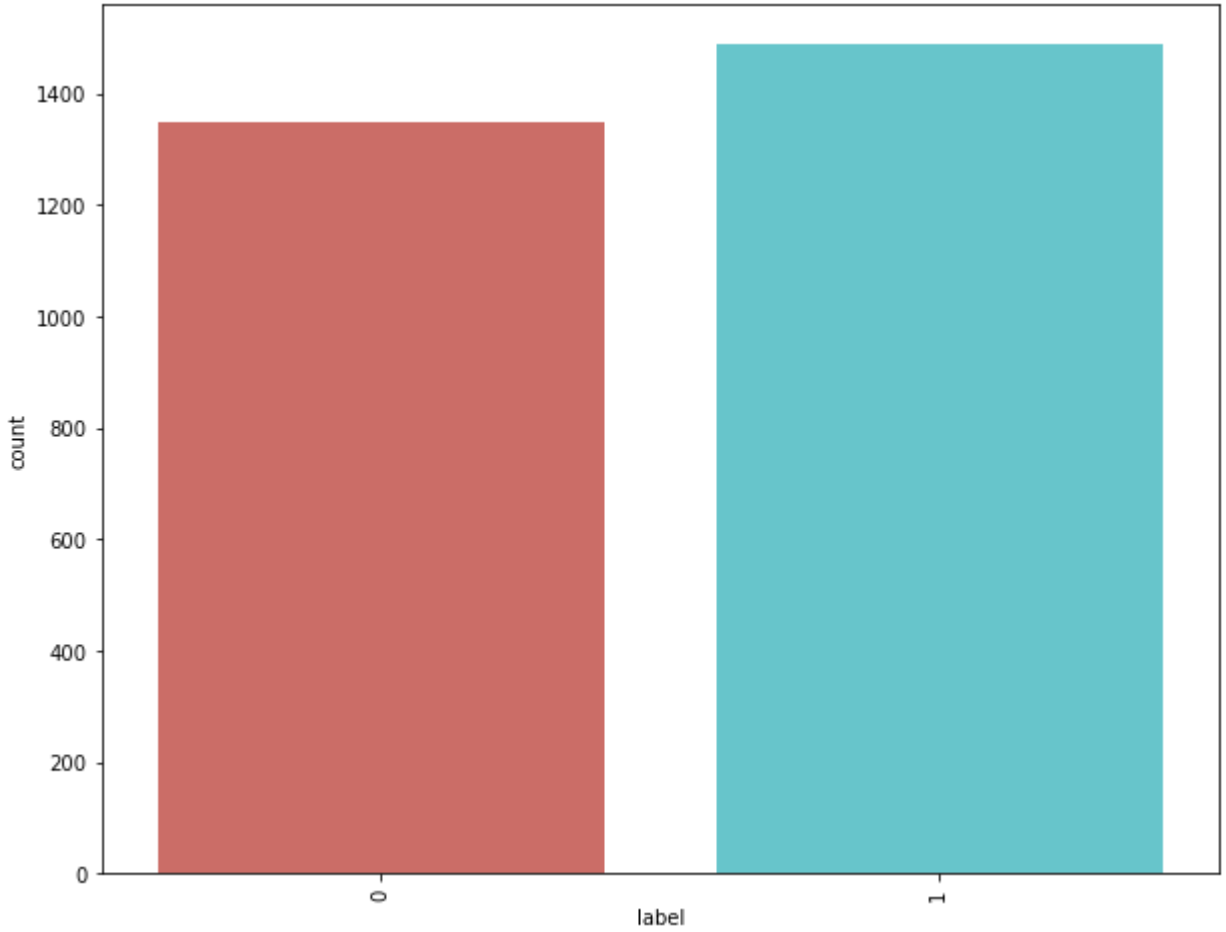
In [10]:
```
# plot the value count visuals
plt.figure(figsize=(15,10))
sns.countplot('subreddit', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [11]:
```
# value count of label
df['label'].value_counts()
```

Out[11]:
```
1    1488
0    1350
Name: label, dtype: int64
```

In [12]:
```
# plot the value count visuals
plt.figure(figsize=(10,8))
sns.countplot('label', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [13]:
```
# import text cleaning libraries
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] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [14]:
```
# define a function to clean text
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
df["text"] = df["text"].apply(clean)
```

In [15]:
```
# plot a wordcloud for more text visualization
# import the libraries
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
text = " ".join(i for i in df.text)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,
background_color="red").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [16]:
```
# apply map to categorical variables in label
# let 0 be equal to stress and 1 be unstress
df["label"] = df["label"].map({0: "Stress", 1: "Unstress"})
df = df[["text", "label"]]
print(df.head())
```

```
                                                text     label
0  said felt way sugget go rest trigger ahead you...  Unstress
1  hey rassist sure right place post goe  im curr...    Stress
2  mom hit newspap shock would know dont like pla...  Unstress
3  met new boyfriend amaz kind sweet good student...  Unstress
4  octob domest violenc awar month domest violenc...  Unstress
```

Features Selection

In [ ]:
```
# transform text to vector
# import the libraries
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
import pickle

# choose dependent and independent variable
x = np.array(df["text"])
y = np.array(df["label"])

# transforming text to vectors
cv = CountVectorizer()
X = cv.fit_transform(x)

# save the pickle file
pickle.dump(X, open("Xcv.pkl", "wb"))
# split into test and train set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

Build the model

In [24]:
```
# import the algorithm library
# model selection
#from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
#from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score,f1_score
# using naive bayes
model = MultinomialNB()
# fit the model to the set
model.fit(x_train, y_train)
```

Out[24]:
```
MultinomialNB()
```

In [25]:
```
# make y prediction
MNB_pred = model.predict(x_test)
```

Model Evaluation and Validation

In [26]:
```
# print classification report
print(classification_report(MNB_pred, y_test))
```

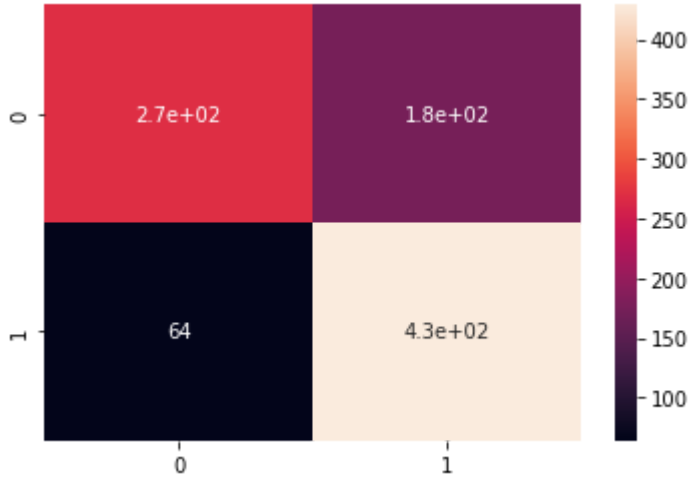
```
              precision    recall  f1-score   support

      Stress       0.61      0.81      0.69       333
    Unstress       0.87      0.71      0.78       604

    accuracy                           0.74       937
   macro avg       0.74      0.76      0.74       937
weighted avg       0.78      0.74      0.75       937
```

In [27]:
```
# print and plot confusion matrix
cm = confusion_matrix(y_test, MNB_pred)
sns.heatmap(cm, annot=True)
```

Out[27]: <AxesSubplot:>



In [22]:
```
# check model accuracy
accuracy = accuracy_score(y_test, MNB_pred)
print("accuracy score is", accuracy)
```

```
accuracy score is 0.7449306296691569
```

Make Predictions with the Model

In [29]:
```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: sushmitha is mentally stable now
['Unstress']
```

In [30]:
```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: this feeling is frustrating
['Stress']
```

In [31]:
```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

```
Enter a Text: oh my Gosh i am so tired
['Stress']
```

In [32]:
```
# save or dump the model in pickle
# load the library
import pickle
pickle.dump(model, open("stress_model.pkl", "wb"))
```

In [ ]:
```

```