In [79]:
```
import pandas as pd
import numpy as np
```

In [80]:
```
# Load Data Viz Pkgs
import seaborn as sns
```

In [81]:
```
# Load Text Cleaning Pkgs
import neattext.functions as nfx
```

In [82]:
```
# Load ML Pkgs
# Estimators
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
# Transformers
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

In [83]:
```
# Load Dataset
df = pd.read_csv("Desktop\emotion_dataset_2.csv")
```

In [84]:
```
df.head()
```

Out[84]:

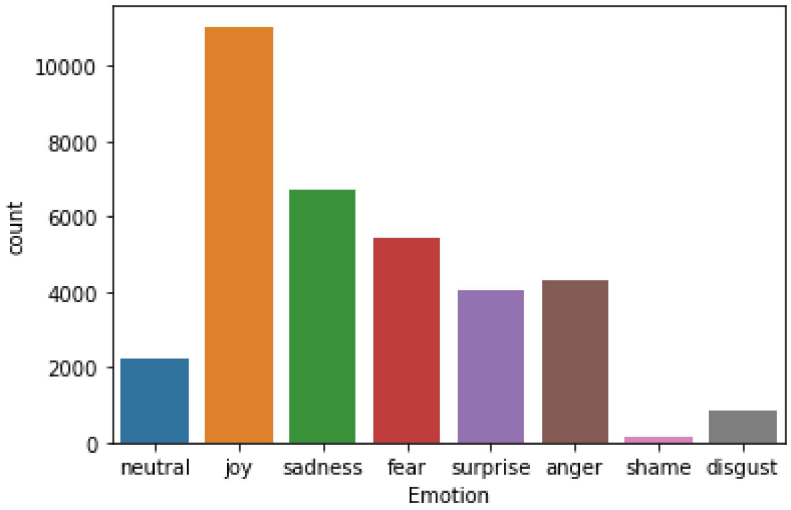
	Unnamed: 0	Emotion	Text	Clean_Text
0	0	neutral	Why ?	NaN
1	1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow
2	2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS SH...
3	3	joy	Such an eye ! The true hazel eye-and so brill...	eye true hazel eyeand brilliant Regular feat...
4	4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe hugggzzz u babe naamazed nga ako e...

In [85]:
```
df['Emotion'].value_counts()
```

Out[85]:
```
joy         11045
sadness      6722
fear         5410
anger        4297
surprise     4062
neutral      2254
disgust       856
shame         146
Name: Emotion, dtype: int64
```

In [86]:
```
sns.countplot(x='Emotion',data=df)
```

Out[86]: <AxesSubplot:xlabel='Emotion', ylabel='count'>



In [87]:
```
dir(nfx)
```

Out[87]:
```
['BTC_ADDRESS_REGEX',
 'CURRENCY_REGEX',
 'CURRENCY_SYMB_REGEX',
 'Counter',
 'DATE_REGEX',
 'EMAIL_REGEX',
 'EMOJI_REGEX',
 'HASTAG_REGEX',
 'MASTERCard_REGEX',
 'MD5_SHA_REGEX',
 'MOST_COMMON_PUNCT_REGEX',
 'NUMBERS_REGEX',
 'PHONE_REGEX',
 'PoBOX_REGEX',
 'SPECIAL_CHARACTERS_REGEX',
 'STOPWORDS',
 'STOPWORDS_de',
 'STOPWORDS_en',
 'STOPWORDS_es',
 'STOPWORDS_fr',
 'STOPWORDS_ru',
 'STOPWORDS_yo',
 'STREET_ADDRESS_REGEX',
 'TextFrame',
 'URL_PATTERN',
 'USER_HANDLES_REGEX',
 'VISACard_REGEX',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__generate_text',
 '__loader__',
 '__name__',
 '__numbers_dict',
 '__package__',
 '__spec__',
 '_lex_richness_herdan',
 '_lex_richness_maas_ttr',
 'clean_text',
 'defaultdict',
 'digit2words',
```

```
 'extract_btc_address',
 'extract_currencies',
 'extract_currency_symbols',
 'extract_dates',
 'extract_emails',
 'extract_emojis',
 'extract_hashtags',
 'extract_html_tags',
 'extract_mastercard_addr',
 'extract_md5sha',
 'extract_numbers',
 'extract_pattern',
 'extract_phone_numbers',
 'extract_postoffice_box',
 'extract_shortwords',
 'extract_special_characters',
 'extract_stopwords',
 'extract_street_address',
 'extract_terms_in_bracket',
 'extract_urls',
 'extract_userhandles',
 'extract_visacard_addr',
 'fix_contractions',
 'generate_sentence',
 'hamming_distance',
 'inverse_df',
 'lexical_richness',
 'markov_chain',
 'math',
 'nlargest',
 'normalize',
 'num2words',
 'random',
 're',
 'read_txt',
 'remove_accents',
 'remove_bad_quotes',
 'remove_btc_address',
 'remove_currencies',
 'remove_currency_symbols',
 'remove_custom_pattern',
 'remove_custom_words',
 'remove_dates',
 'remove_emails',
 'remove_emojis',
 'remove_hashtags',
 'remove_html_tags',
 'remove_mastercard_addr',
 'remove_md5sha',
 'remove_multiple_spaces',
 'remove_non_ascii',
 'remove_numbers',
 'remove_phone_numbers',
 'remove_postoffice_box',
 'remove_puncts',
 'remove_punctuations',
 'remove_shortwords',
 'remove_special_characters',
 'remove_stopwords',
 'remove_street_address',
 'remove_terms_in_bracket',
 'remove_urls',
 'remove_userhandles',
 'remove_visacard_addr',
```

```
'replace_bad_quotes',
'replace_currencies',
'replace_currency_symbols',
'replace_dates',
'replace_emails',
'replace_emojis',
'replace_numbers',
'replace_phone_numbers',
'replace_special_characters',
'replace_term',
'replace_urls',
'string',
'term_freq',
'to_txt',
'unicodedata',
'word_freq',
'word_length_freq']
```

In [88]:
```
df['Clean_Text'] = df['Text'].apply(nfx.remove_userhandles)
```

In [89]:
```
# Stopwords
df['Clean_Text'] = df['Clean_Text'].apply(nfx.remove_stopwords)
```

In [90]:
```
# Features & Labels
Xfeatures = df['Clean_Text']
ylabels = df['Emotion']
```

In [91]:
```
#  Split Data
x_train,x_test,y_train,y_test = train_test_split(Xfeatures,ylabels,test_size=0.3,ran
# Build Pipeline
```

In [92]:
```
from sklearn.pipeline import Pipeline
```

In [93]:
```
# LogisticRegression Pipeline
pipe_lr = Pipeline(steps=[('cv',CountVectorizer()),('lr',LogisticRegression())])
```

In [94]:
```
# Train and Fit Data
pipe_lr.fit(x_train,y_train)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[94]:
```
Pipeline(steps=[('cv', CountVectorizer()), ('lr', LogisticRegression())])
```

In [95]:
```
pipe_lr
```

Out[95]:
```
Pipeline(steps=[('cv', CountVectorizer()), ('lr', LogisticRegression())])
```

In [96]:
```
pipe_lr.score(x_test,y_test)
```

Out[96]:
```
0.6187967043494922
```

In [102…
```
ex1 = "I'm enjoying here"
```

In [98]:
```
pipe_lr.predict([ex1])
```

Out[98]:
```
array(['joy'], dtype=object)
```

In [99]:
```
# Prediction Prob
pipe_lr.predict_proba([ex1])
```

Out[99]:
```
array([[0.04870954, 0.02610009, 0.10138903, 0.46511905, 0.20264447,
        0.08458137, 0.00259221, 0.06886424]])
```

In [100…
```
# To Know the classes
pipe_lr.classes_
```

Out[100…
```
array(['anger', 'disgust', 'fear', 'joy', 'neutral', 'sadness', 'shame',
       'surprise'], dtype=object)
```

In [101…
```
# Save Model & Pipeline
import joblib
pipeline_file = open("emotion_classifier_pipe_lr_03_june_2021.pkl","wb")
joblib.dump(pipe_lr,pipeline_file)
pipeline_file.close()
```

In [ ]:

In [ ]: