End 2 End NLP Project

- Emotion Detection In Text
- Text Classifier

In [1]:
```
# Load EDA Pkgs
import pandas as pd
import numpy as np
```

In [2]:
```
# Load Data Viz Pkgs
import seaborn as sns
```

In [3]:
```
# Load Text Cleaning Pkgs
import neattext.functions as nfx
```

In [4]:
```
# 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 [5]:
```
# Load Dataset
df = pd.read_csv("data/emotion_dataset_raw.csv")
```

In [6]:
```
df.head()
```

Out[6]:

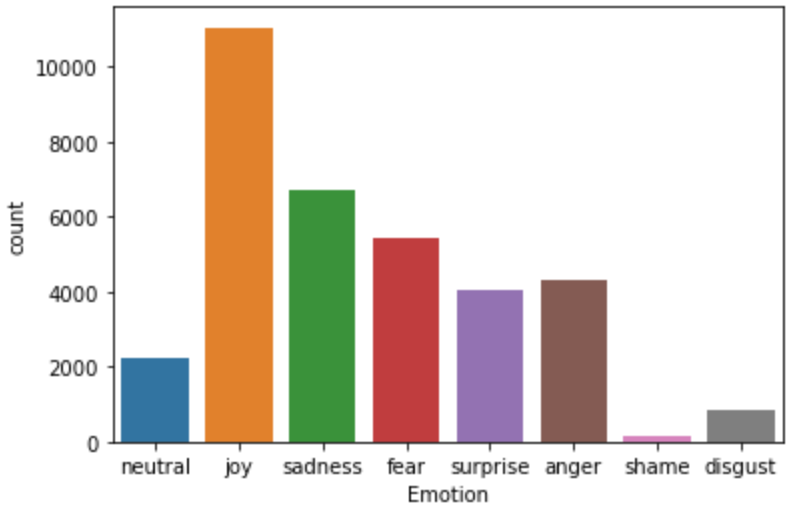
	Emotion	Text
0	neutral	Why ?
1	joy	Sage Act upgrade on my to do list for tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...
3	joy	Such an eye ! The true hazel eye-and so brill...
4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...

In [7]:
```
# Value Counts
df['Emotion'].value_counts()
```

Out[7]:
```
joy         11045
sadness      6722
fear         5410
anger        4297
surprise     4062
neutral      2254
disgust       856
shame         146
Name: Emotion, dtype: int64
```

In [8]:
```
# Plot
sns.countplot(x='Emotion',data=df)
```

Out[8]:
```
<AxesSubplot:xlabel='Emotion', ylabel='count'>
```



In [9]:
```
# Data Cleaning
dir(nfx)
```

Out[9]:
```
['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 [10]:
```
# User handles
df['Clean_Text'] = df['Text'].apply(nfx.remove_userhandles)
```

In [11]:
```
# Stopwords
df['Clean_Text'] = df['Clean_Text'].apply(nfx.remove_stopwords)
```

In [12]:
```
df
```

Out[12]:

	Emotion	Text	Clean_Text
0	neutral	Why ?	?
1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL!!! MAN HATE FUNERALS...
3	joy	Such an eye ! The true hazel eye-and so brill...	eye ! true hazel eye-and brilliant ! Regular f...
4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe.. hugggzzz u .! babe naamazed nga ako...
...	...	...	...
34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift! Hope like it! hand wear ! It'll warm! Lol
34788	joy	The world didnt give it to me..so the world MO...	world didnt me..so world DEFINITELY cnt away!!!
34789	anger	A man robbed me today .	man robbed today .
34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY, #Losing YOU...
34791	sadness	I think about you baby, and I dream about you ...	think baby, dream time

34792 rows × 3 columns

In [13]:
```
# Features & Labels
Xfeatures = df['Clean_Text']
ylabels = df['Emotion']
```

In [14]:
```
#  Split Data
x_train,x_test,y_train,y_test = train_test_split(Xfeatures,ylabels,test_size=0.3,random_
```

In [15]:
```
# Build Pipeline
from sklearn.pipeline import Pipeline
```

In [16]:
```
# LogisticRegression Pipeline
pipe_lr = Pipeline(steps=[('cv',CountVectorizer()),('lr',LogisticRegression())])
```

In [17]:
```
# Train and Fit Data
pipe_lr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Conver
genceWarning: 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[17]:
```
Pipeline(steps=[('cv', CountVectorizer()), ('lr', LogisticRegression())])
```

In [18]:
```
pipe_lr
```

Out[18]:
```
Pipeline(steps=[('cv', CountVectorizer()), ('lr', LogisticRegression())])
```

In [19]:
```
# Check Accuracy
pipe_lr.score(x_test,y_test)
```

```
0.8934466617393446
```

In [22]:
```
# Make A Prediction
ex1 = "This book was so interesting it made me happy"
```

In [23]:
```
pipe_lr.predict([ex1])
```

Out[23]:
```
array(['joy'], dtype=object)
```

In [24]:
```
# Prediction Prob
pipe_lr.predict_proba([ex1])
```

Out[24]:
```
array([[1.60395969e-03, 7.06093710e-03, 6.95877957e-03, 9.43790359e-01,
        1.00433761e-04, 2.63461456e-02, 6.64779798e-05, 1.40729072e-02]])
```

In [25]:
```
# To Know the classes
pipe_lr.classes_
```

Out[25]:
```
array(['anger', 'disgust', 'fear', 'joy', 'neutral', 'sadness', 'shame',
       'surprise'], dtype=object)
```

In [26]:
```
# 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 [ ]:
```
```