Dataset:

https://data.world/crowdflower/sentiment-analysis-in-text

In [ ]:

```
import pandas as pd
# Reading the csv dataset into a pandas dataframe
df = pd.read_csv('E:/Internships/TCS-iON/Code/MyCode/Tweets/text_emotion.csv', encoding = '
# Adding a column representing 1 for positive and 0 for negative sentiments
df['senti'] = df.apply(lambda x: 1 if (x['sentiment'] == 'enthusiasm' or x['sentiment'] ==
# Deleting unnecessary columns
df = df.drop(['tweet_id', 'sentiment', 'author'], axis = 1)
# Converting the data type to string
df['content'] = df["content"].astype("str")
# Converting all text to lowercase for use
df['content'] = df['content'].str.lower()
df.head()
```

	review	senti
	@tiffanylue i know i was listenin to bad habi...	0
	layin n bed with a headache ughhhh...waitin o...	0
	funeral ceremony...gloomy friday...	0
	wants to hang out with friends soon!	1
	@dannycastillo we want to trade with someone w...	0

i am going to start reading the harry potter series again because that is one awesome story.

In [ ]:

```
import re
import string
from nltk import WordNetLemmatizer
from nltk.stem.snowball import SnowballStemmer
from nltk.corpus import stopwords
# Initialising the nltk stop_words, stemmer and lemmatizer functions
stop_words = set(stopwords.words("english"))
lemmatizer = WordNetLemmatizer()
stemmer = SnowballStemmer("english")
# Creating a function for text cleaning
def textCleanser(myText):
    # Converting each tweet to string
    myText = str(myText)
    # Removing the name titles and the period symbols after it
    myText = re.sub(r'[mdsr]r(s)?\.', '', myText)
    # Removing the '@username' mentions
    myText = re.sub(r'@\w+\s', '', myText)
    # Removing punctuation
    myPunct = string.punctuation
    punctToSpace = str.maketrans(myPunct, len(myPunct)*' ')
    myText = myText.translate(punctToSpace)
    # Removing urls
    myText = re.sub(r'((http(s?)?):/?)(www\.?).+\.com', '', myText)
    myText = re.sub(r'http(s?))', '', myText)
    # Removing numbers
    myText = re.sub(r'\d+', '', myText)
    # Removing stopwords
    myText = [word for word in myText.split(' ') if not word in stop_words]
    myText = [word for word in myText if word != '']
    # Lemmatizing the text
    myText = [lemmatizer.lemmatize(token) for token in myText]
    # Stemming the text
    # myText = [stemmer.stem(token) for token in myText]
    return myText
for i in range(len(df['content'])):
    df['content'][i] = textCleanser(df['content'][i])
df.head()
```

	review	senti
	[know, listenin, bad, habit, earlier, started,...	0
	[layin, n, bed, headache, ughhhh, waitin, call]	0
	[funeral, ceremony, gloomy, friday]	0
	[want, hang, friend, soon]	1
	[want, trade, someone, houston, ticket, one]	0

['going', 'start', 'reading', 'harry', 'potter', 'series', 'one', 'awesome', 'story']

In [ ]:

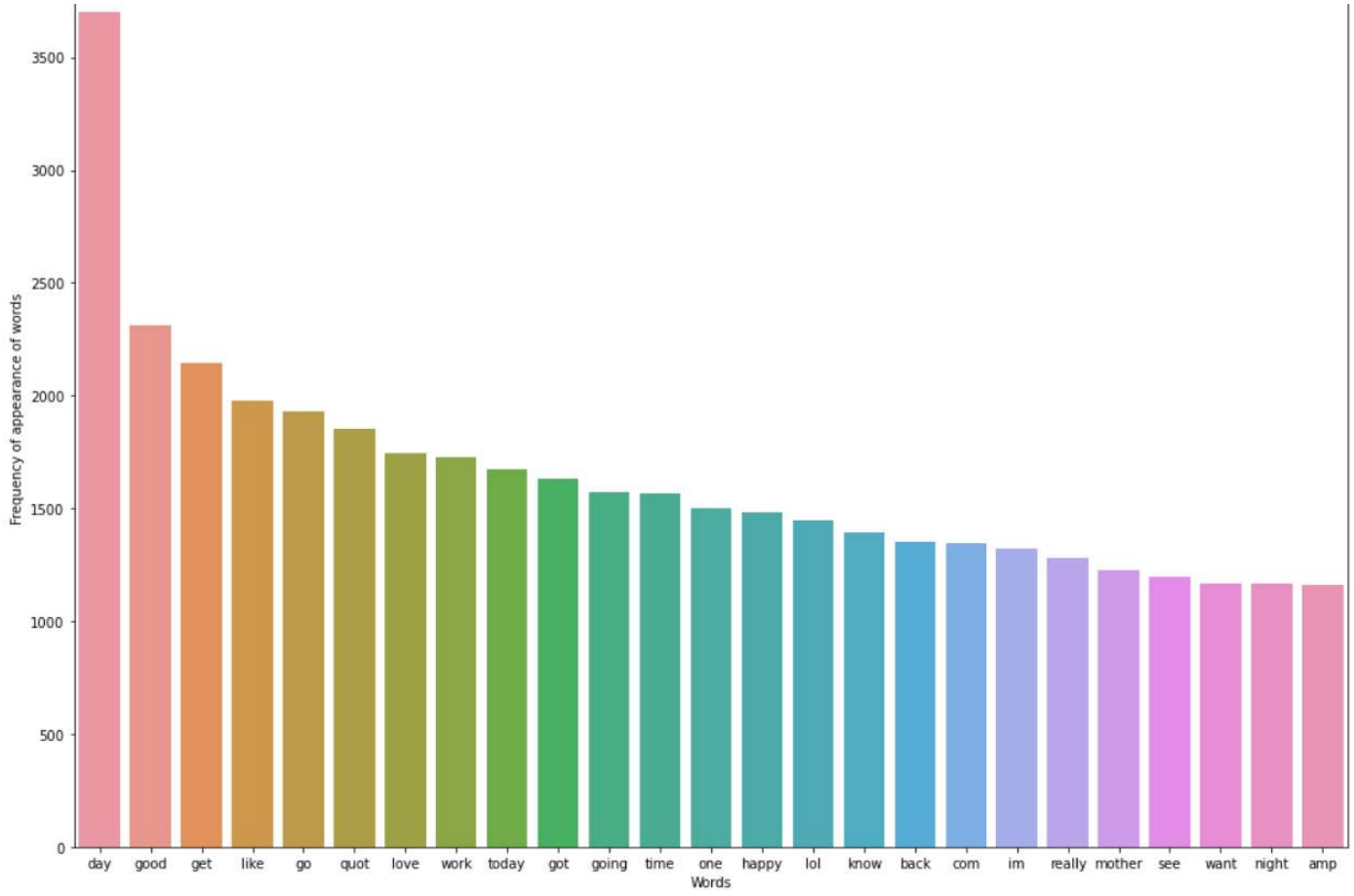
```
myReviews = []
for i in range(len(df['content'])):
    for j in df['content'][i]:
        if j != 'br' and j != 'http':
            myReviews.append(j)
```

In [ ]:

```
from collections import Counter
import collections
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
import numpy as np
np.random.seed(1234)
# Initialising the Count Vectorizer
cv = CountVectorizer()
myBow = cv.fit_transform(myReviews)
wordFrequency = dict(zip(cv.get_feature_names(), np.asarray(myBow.sum(axis = 0)).ravel()))
wordCounter = collections.Counter(wordFrequency)
# Storing the frequency of appearance of words
dfWordCounter = pd.DataFrame(wordCounter.most_common(25), columns = ['word', 'frequency'])
```

In [ ]:

```
# Plotting the top 25 most frequently occuring words
plt.close('all')
fig, ax = plt.subplots(figsize = (17, 12))
sns.barplot(x = 'word', y = 'frequency', data = dfWordCounter, ax = ax)
sns.set_palette('pastel')
plt.xlabel('Words')
plt.ylabel('Frequency of appearance of words')
plt.show()
```



In [ ]:

```
# Defining a dummy function for the tokenizer and preprocessor inputs of the vectorizers so
def dummy_fun(doc):
    return doc
tfidf = TfidfVectorizer(sublinear_tf=True, max_df = 2000, min_df = 50, norm='l2', encoding=
features = []
features = tfidf.fit_transform(df.content).toarray()
labels = df.senti
print(features.shape)
```

Features Shape:

```
(40000, 823)
```

In [ ]:

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
# Splitting the dataframe to training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(df["review"], df['senti'],test_size=0.2
vectorizerCount = CountVectorizer(tokenizer = dummy_fun, preprocessor = dummy_fun)
X_train_counts = vectorizerCount.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
nBayes = MultinomialNB().fit(X_train_tfidf, y_train)
```

In [ ]:

```
y_= nBayes.predict(vectorizerCount.transform(X_test))
```

In [ ]:

```
# Testing the data
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_,normalize=True))
```

Accuracy:

```
71.31%
```