In [1]:

```
from EmojiSentiWordnet.sentiwordnet import *
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import nltk
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import lightgbm as lgb
from scipy.sparse import coo_matrix, hstack
from sklearn.metrics import accuracy_score
wnl = WordNetLemmatizer()
stopwords = nltk.corpus.stopwords.words('english')
```

In [2]:

```
esw=mojiSentiWordnet()
cry_synset=esw.synset(lemma='😭')
```

Here we have 127 emoji icons in the wordnet, can be more if we have more rich dataset. For each emoji character we can get a positive score, a negative score and occurance probability

In [3]:

```
print(f'The positive score of 😭:{cry_synset.get_pscore()}')
print(f'The negative score of 😭:{cry_synset.get_nscore()}')
print(f'The occurance rate of 😭:{cry_synset.get_occ()}')
print(f'The unicode id of 😭:{cry_synset.get_id()}')
print(f'The description of 😭:{cry_synset.get_des()}')
```

```
The positive score of 😭:0.019443217
The negative score of 😭:0.980556783
The occurance rate of 😭:0.0511670928
The unicode id of 😭:1F62D
The description of 😭::loudly_crying_face:
```

Here we test the tool on a sample dataset

In [4]:

```
tweet_df=pd.read_csv('tweet_senti.txt',sep=',')
tweet_df.head()
```

Out[4]:

	tweets	labels
0	lmfaoo 😭 😭 😭 😭 😭	0
1	i hate this feeling 😢	0
2	can't believe i just went out in this cold to ...	0
3	i need a new trap house, so if you really fuck...	0
4	<user> so very sorry for your loss. 💔	0

In [5]:

```
def clean_tweet(text):
    text_token=word_tokenize(text)
    text_cleaned=[]
    for t in text_token :
        t=t.lower()
        if len(t)>1 and (t not in stopwords):
            text_cleaned.append(wnl.lemmatize(t))
    return ' '.join(text_cleaned)
```

In [6]:
```
text='i need a new trap house, so if you really fuck 😢'
clean_tweet(text)
```

Out[6]:
```
'need new trap house really fuck'
```

In [7]:
```
tweet_df["cleaned_tweets"]=tweet_df["tweets"].apply(clean_tweet)
```

In [8]:
```
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), lowercase=True, use_idf=T
tweet_tfidf=tfidf.fit_transform(tweet_df["cleaned_tweets"])
```

In [9]:
```
tweets=tweet_df["tweets"]
p_scores=[]
n_scores=[]
for tweet in tweets:
    p_score=0
    n_score=0
    tweet_emoji=set(adv.extract_emoji(tweet)['emoji_flat'])
    for emoji_icon in tweet_emoji:
        if emoji_icon in esw.dict.keys():
            emoji_synset=esw.synset(lemma=emoji_icon)
            p_score+=emoji_synset.get_pscore()
            n_score+=emoji_synset.get_nscore()
    p_scores.append(p_score)
    n_scores.append(n_score)
p_scores=np.array([p_scores]).T
n_scores=np.array([n_scores]).T
```

In [10]:
```
n_scores.shape
```

Out[10]:
```
(13200, 1)
```

In [11]:
```
X_1=tweet_tfidf
X_2=hstack([tweet_tfidf,coo_matrix(p_scores),coo_matrix(n_scores)])
Y=tweet_df["labels"]
```

In [12]:
```
X_2.shape
```

Out[12]:
```
(13200, 65798)
```

Without Emoji Sentiment Score

In [13]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X_1, Y, test_size=0.1)
lr = LogisticRegression(n_jobs=-1)
lr.fit(X_train, Y_train)
ypred=lr.predict(X_test)
print('Accuracy on dataset:', accuracy_score(ypred, Y_test))
print('\n')
```

```
Accuracy on dataset: 0.896969696969697
```

With Emoji Sentiment Score

In [14]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X_2, Y, test_size=0.1)
lr = LogisticRegression(n_jobs=-1)
lr.fit(X_train, Y_train)
ypred=lr.predict(X_test)
print(’Accuracy on dataset:’, accuracy_score(ypred, Y_test))
print(’\n’)
```

```
Accuracy on dataset: 0.9598484848484848
```

In [ ]: