Import modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
import warnings
%matplotlib inline

warnings.filterwarnings('ignore')
```

Loading the dataset

In [4]:
```
df = pd.read_csv(r"C:\Users\dell\Downloads\Twitter Sentiments.csv")
```

In [5]:
```
df.head()
```

Out[5]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      31962 non-null  int64 
 1   label   31962 non-null  int64 
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

Preprocessing the dataset

In [41]:
```
# removes pattern in the input text
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt
```

In [42]:
```
# remove twitter handles (@user)
df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")
```

In [43]:
```
df.head()
```

Out[43]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide: society now #motivation

In [44]:
```
# remove special characters, numbers and punctuations
df['clean_tweet'] = df['clean_tweet'].str.replace("[^a-zA-Z#]", " ")
df.head()
```

Out[44]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide: society now #motivation

In [45]:
```
# remove short words
df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x.split(
df.head()
```

Out[45]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit can't cause they don't off...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model love take with time urð±!!! ðð...
4	5	0	factsguide: society now #motivation	factsguide: society #motivation

In [46]:
```python
# individual words considered as tokens
tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

Out[46]:
```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, can't, cause, they, do...
2                              [bihday, your, majesty]
3    [#model, love, take, with, time, urð±!!!, ð...
4                  [factsguide:, society, #motivation]
Name: clean_tweet, dtype: object
```

In [47]:
```python
# stem the words
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda sentence: [stemmer.stem(word) for wo
tokenized_tweet.head()
```

Out[47]:
```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, can't, caus, they, don'...
2                              [bihday, your, majesti]
3    [#model, love, take, with, time, urð±!!!, ð...
4                       [factsguide:, societi, #motiv]
Name: clean_tweet, dtype: object
```

In [48]:
```python
# combine words into single sentence
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

df['clean_tweet'] = tokenized_tweet
df.head()
```

Out[48]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit can't caus they don't offer...
2	3	0	bihday your majesty	bihday your majesti
3	4	0	#model i love u take with u all the time in ...	#model love take with time urð±!!! ðð...
4	5	0	factsguide: society now #motivation	factsguide: societi #motiv

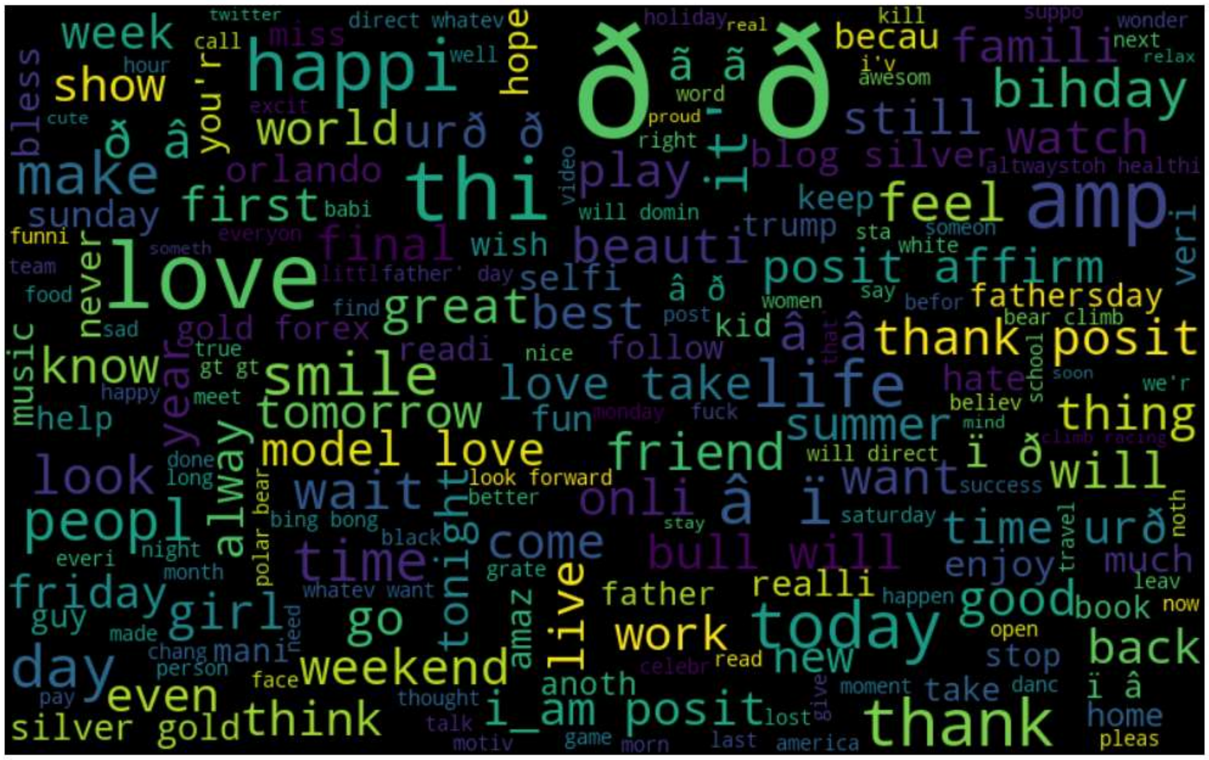
Exploratory Data Analysis

In [21]:

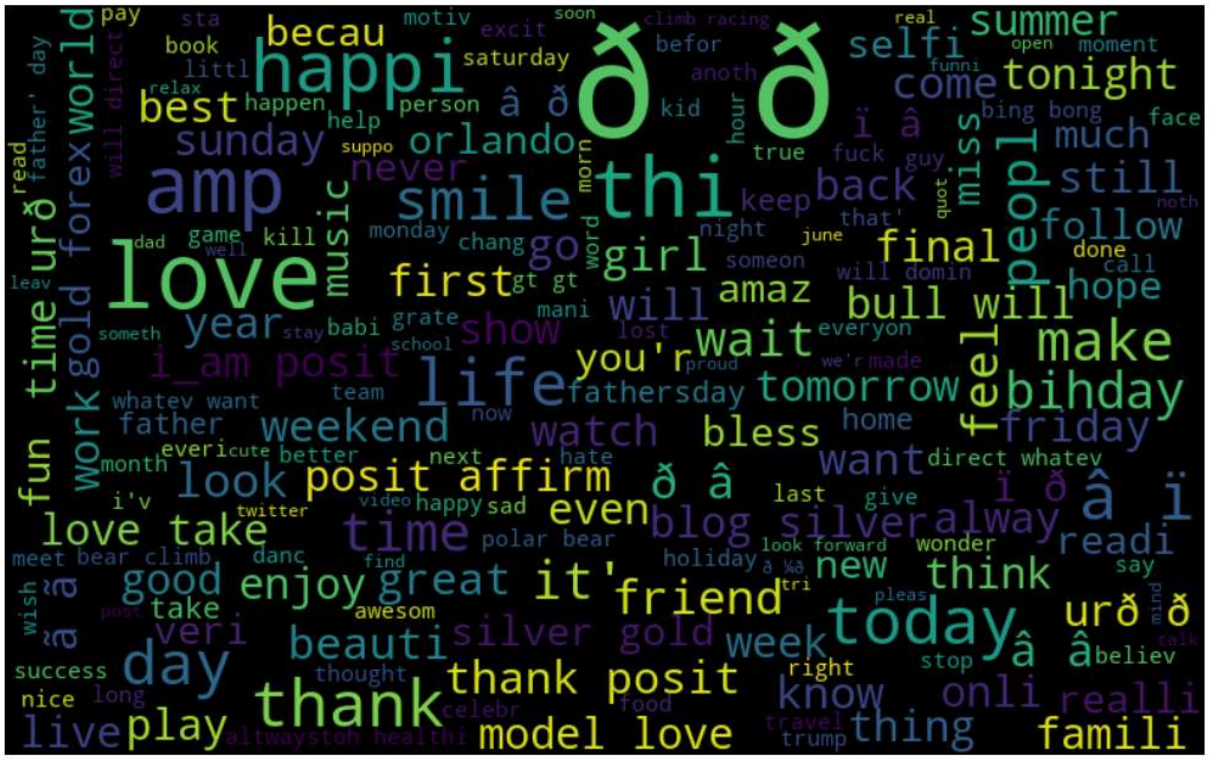
```
#!pip install wordcloud
```

In [49]:

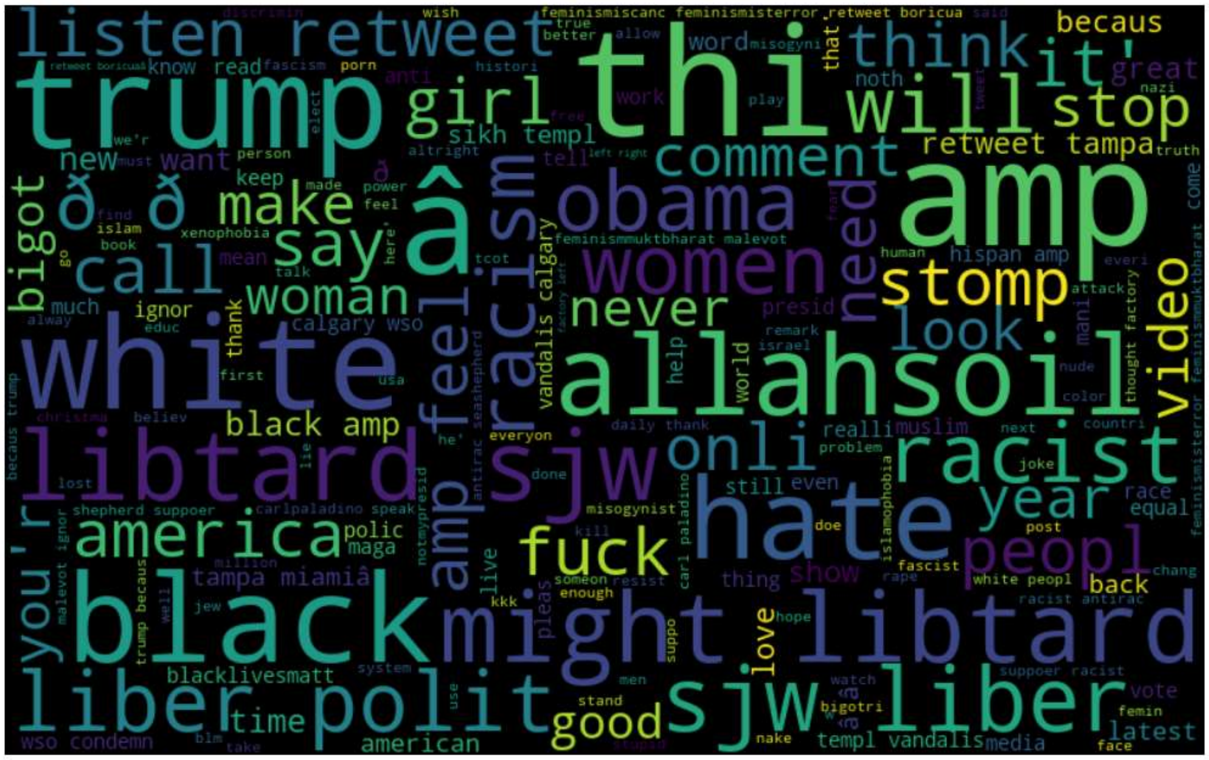
```
# visualize the frequent words
all_words = " ".join([sentence for sentence in df['clean_tweet']])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).ge

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [50]:
```
# frequent words visualization for +ve
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==0]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).ge

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [51]:
```
# frequent words visualization for -ve
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).ge

plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



```
In [52]: # extract the hashtag
         def hashtag_extract(tweets):
             hashtags = []
             # loop words in the tweet
             for tweet in tweets:
                 ht = re.findall(r"#(\w+)" ,tweet)
                 hashtags.append(ht)
             return hashtags
```

```
In [53]: # extract hashtags from non-racist/sexist tweets
         ht_positive = hashtag_extract(df['clean_tweet'][df['label']==0])

         # extract hashtags from racist/sexist tweets
         ht_negative = hashtag_extract(df['clean_tweet'][df['label']==1])
```

```
In [54]: ht_positive[:5]
```

```
Out[54]: [['run'], ['lyft', 'disapoint', 'getthank'], [], ['model'], ['motiv']]
```

```
In [55]: # unnest list
         ht_positive = sum(ht_positive, [])
         ht_negative = sum(ht_negative, [])
```

```
In [56]: ht_positive[:5]
```

```
Out[56]: ['run', 'lyft', 'disapoint', 'getthank', 'model']
```

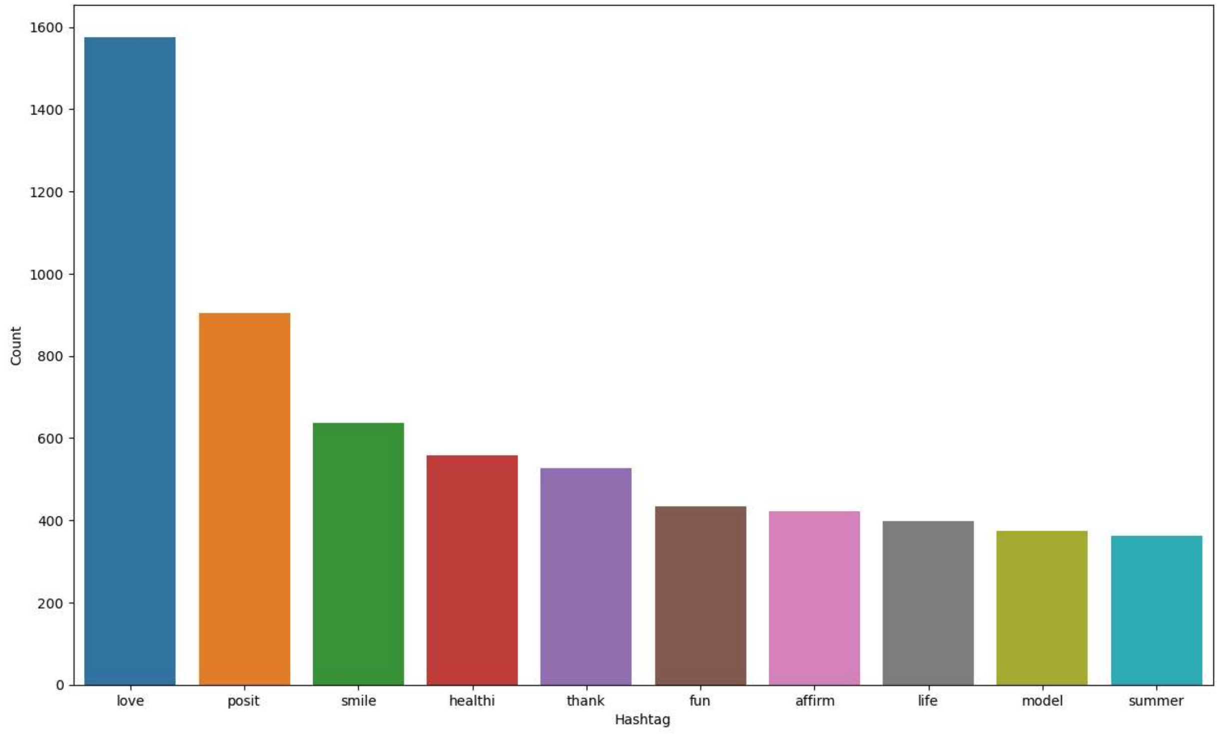
```
In [57]: freq = nltk.FreqDist(ht_positive)
         d = pd.DataFrame({'Hashtag': list(freq.keys()),
                 'Count': list(freq.values())})
d.head()
```

Out[57]:

	Hashtag	Count
0	run	70
1	lyft	2
2	disapoint	1
3	getthank	2
4	model	374

In [58]:

```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```



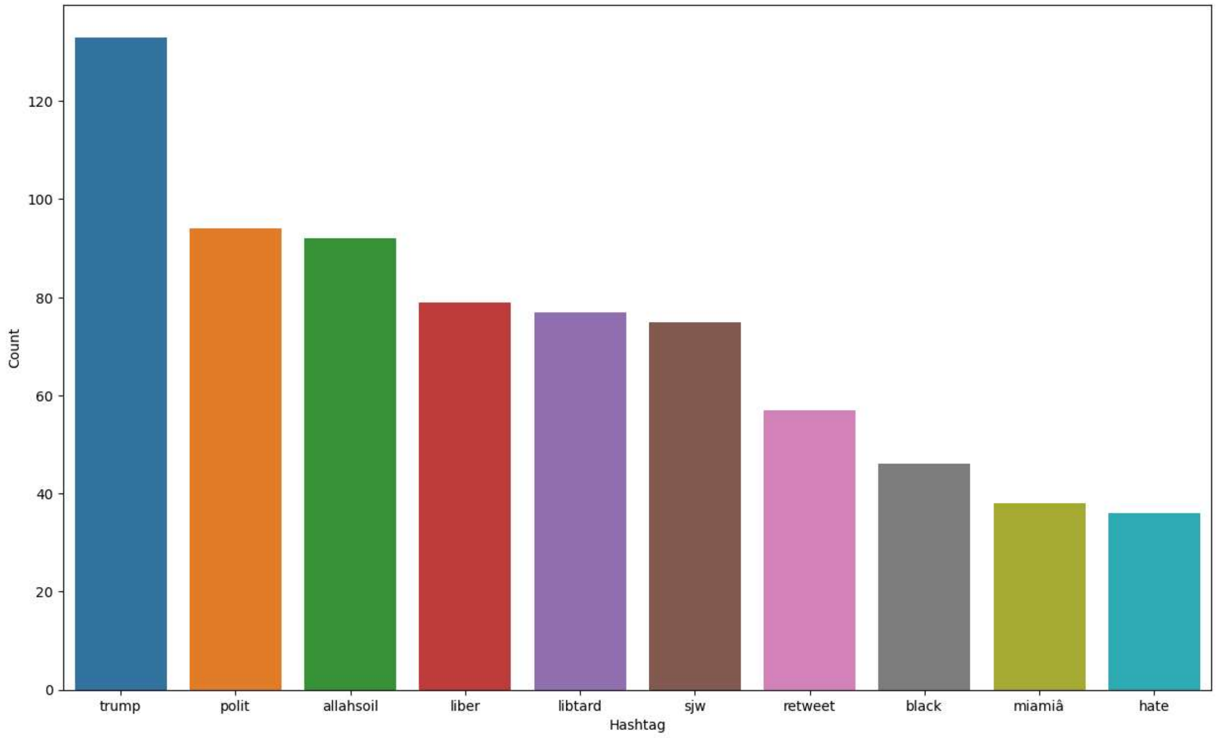
In [59]:

```
freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                 'Count': list(freq.values())})
d.head()
```

Out[59]:

	Hashtag	Count
0	cnn	9
1	michigan	2
2	tcot	14
3	australia	6
4	opkillingbay	2

In [60]:
```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```



Input Split

In [61]:
```
# feature extraction
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_wor
bow = bow_vectorizer.fit_transform(df['clean_tweet'])
```

In [62]:
```
# bow[0].toarray()
```

In [63]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bow, df['label'], random_state=
```

Model Training

```
In [64]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import f1_score, accuracy_score
```

```
In [65]: # training
         model = LogisticRegression()
         model.fit(x_train, y_train)
```

```
Out[65]: ▾ LogisticRegression

         LogisticRegression()
```

```
In [66]: # testing
         pred = model.predict(x_test)
         f1_score(y_test, pred)
```

```
Out[66]: 0.5083135391923991
```

```
In [67]: accuracy_score(y_test,pred)
```

```
Out[67]: 0.9481917156801402
```

```
In [70]: # use probability to get output
         pred_prob = model.predict_proba(x_test)
         pred = pred_prob[:, 1] >= 0.3
         pred = pred.astype(int)

         f1_score(y_test, pred)
```

```
Out[70]: 0.560856864654333
```

```
In [71]: accuracy_score(y_test,pred)
```

```
Out[71]: 0.9435615066950319
```

```
In [75]: pred_prob[0][1] >= 0.3
```

```
Out[75]: False
```

```
In [ ]:
```