Importing Required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import re
import string
import nltk
import warnings
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score
from wordcloud import WordCloud
warnings.simplefilter('ignore')
```

Loading and Inspecting Data

In [2]:
```
df = pd.read_csv('Twitter Sentiments.csv')
df.head()
```

Out[2]:

	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 [3]:
```
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
```

In [4]:
```
df.shape
```

Out[4]:
```
(31962, 3)
```

Data Preprocessing

In [5]:
```
def remove_pattern(input_text, pattern):
    r = re.findall(pattern, input_text)
    for word in r:
        input_text = re.sub(word, '', input_text)
    return input_text

df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], '@[\w]*')
```

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

Out[6]:

	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 [7]:
```
df.shape
```

Out[7]:
```
(31962, 4)
```

In [8]:
```
df['clean_tweet'] = df['clean_tweet'].str.replace('[^a-zA-Z#]', ' ', regex=
```

In [9]:
```
df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x
```

Tokenization and Stemming

In [10]:
```
tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

Out[10]:
```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                               [bihday, your, majesty]
3                     [#model, love, take, with, time]
4                   [factsguide, society, #motivation]
Name: clean_tweet, dtype: object
```

In [11]:
```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda sentence: [stemmer.stem(word)
tokenized_tweet.head()
```

Out[11]:
```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                                [bihday, your, majesti]
3                     [#model, love, take, with, time]
4                         [factsguid, societi, #motiv]
Name: clean_tweet, dtype: object
```

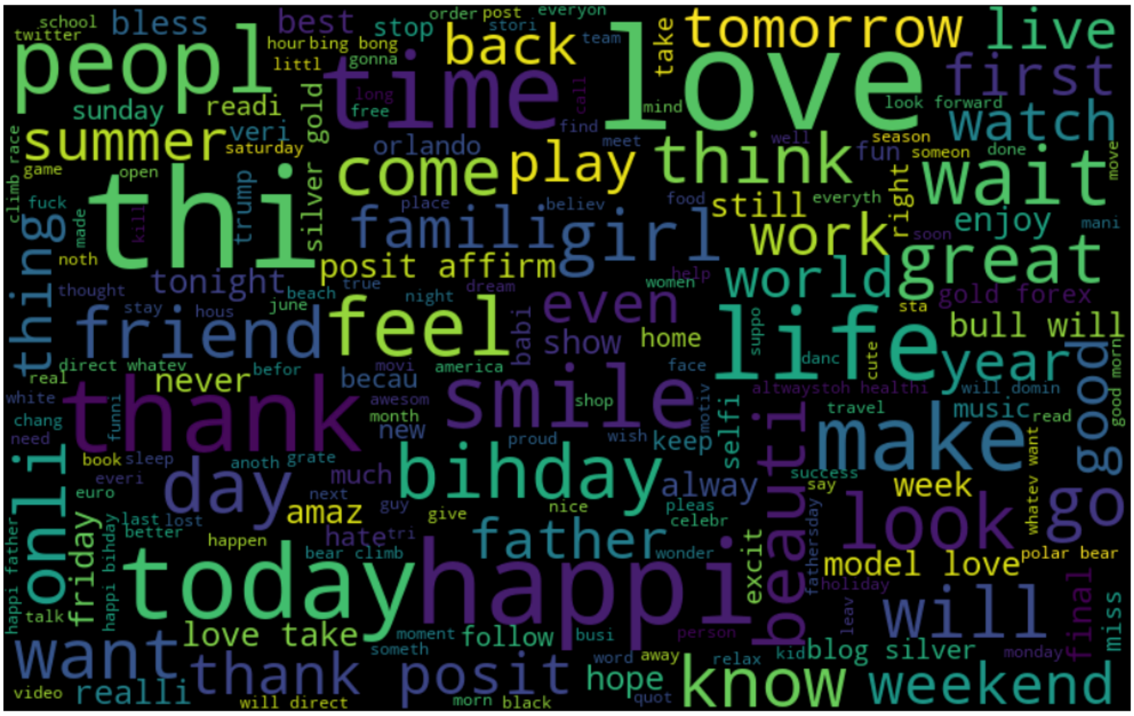
In [12]:
```
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
df['clean_tweet'] = tokenized_tweet
df.head()
```

Out[12]:

	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 caus they offer wheelchair ...
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
4	5	0	factsguide: society now #motivation	factsguid societi #motiv

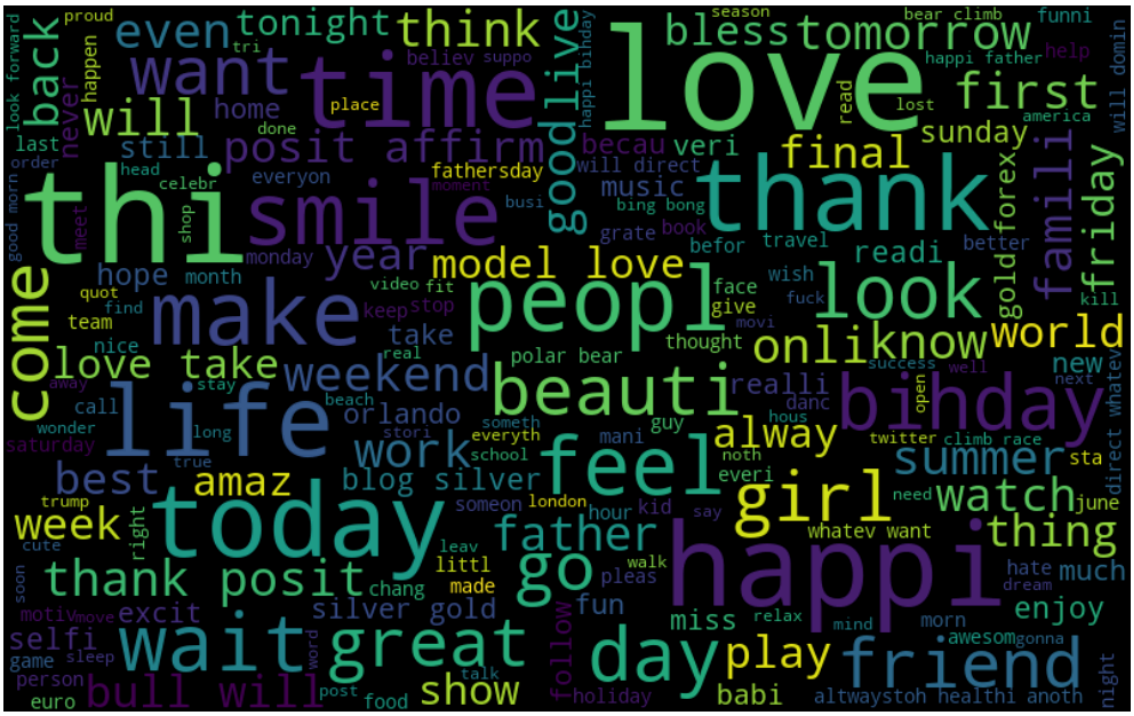
Word Cloud for All Tweets (Frequent Words Visualization)

In [13]:
```
# 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=

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



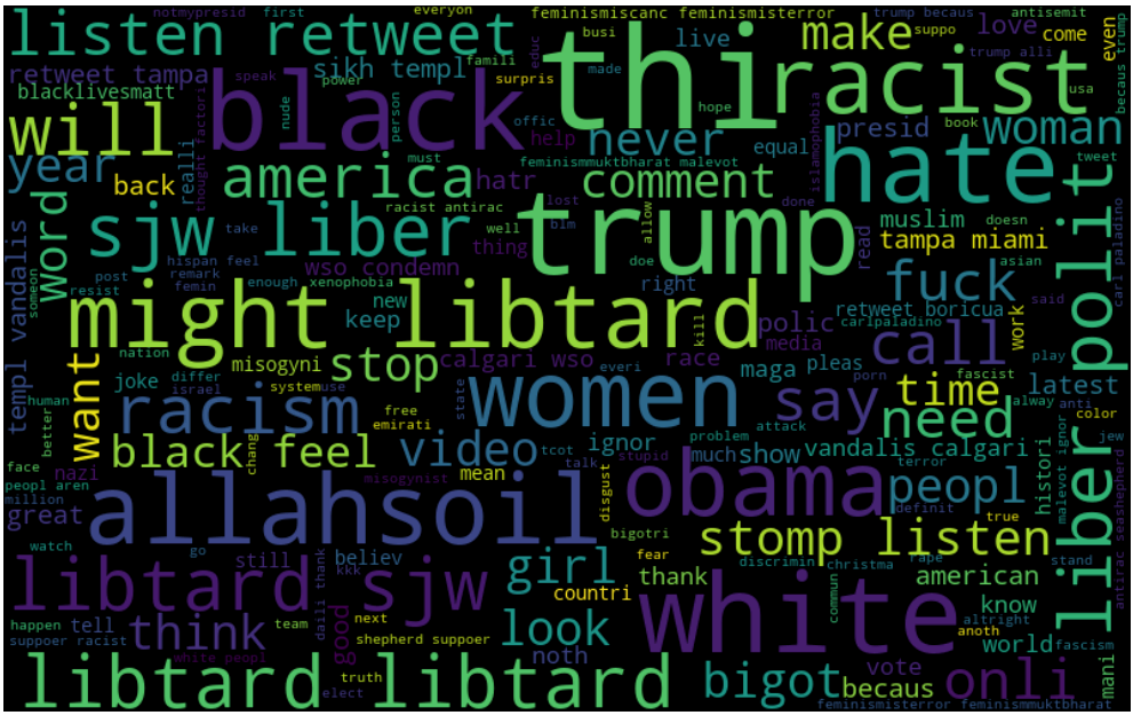
Word Cloud for Positive Tweets (Frequent Words Visualization for Positive Tweets)

In [14]:
```
# frequent words visualization for +ve
# Visualize the frequent words
all_words=' '.join([sentence for sentence in df['clean_tweet'][df['label']==
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=

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



Word Cloud for Negative Tweets (Frequent Words Visualization for Negative Tweets)

In [15]:
```
# frequent words visualization for -ve
# Visualize the frequent words
all_words=' '.join([sentence for sentence in df['clean_tweet'][df['label']==
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=

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



Hashtag Extraction

In [16]:
```
import re

def hastag_extract(tweets):
    hashtags = []
    # Loop over each tweet
    for tweet in tweets:
        ht = re.findall(r'#(\w+)', tweet)
        hashtags.extend(ht)  # use extend to avoid nested lists
    return hashtags
```

In [17]:
```
# extract hastag from no racist/sexist tweets
ht_positive=hastag_extract(df['clean_tweet'][df['label']==0])
#extract hastag from racist/sexist tweets
ht_negative = hastag_extract(df['clean_tweet'][df['label'] == 1])
```

In [18]:
```
ht_positive[:5]
```

Out[18]:
```
['run', 'lyft', 'disapoint', 'getthank', 'model']
```

In [19]:
```
ht_negative[:5]
```

Out[19]:
```
['cnn', 'michigan', 'tcot', 'australia', 'opkillingbay']
```

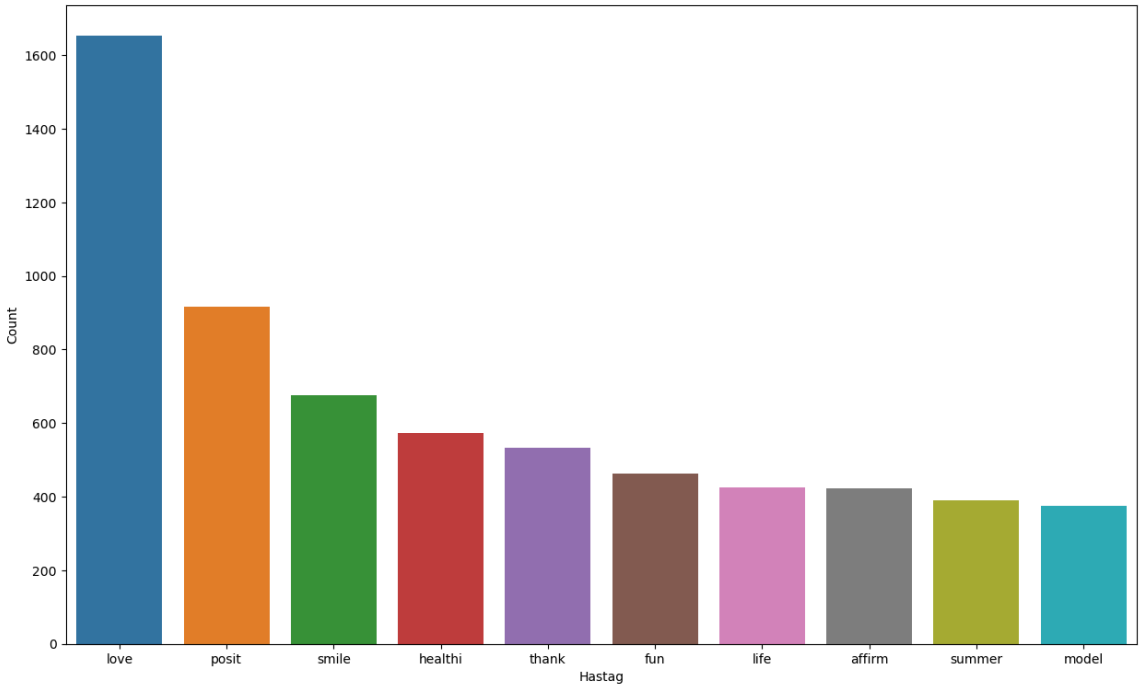
In [20]:
```
freq=nltk.FreqDist(ht_positive)
d=pd.DataFrame({'Hastag': list(freq.keys()),
                                'Count':list(freq.values())})
d.head()
```

Out[20]:

	Hastag	Count
0	run	72
1	lyft	2
2	disapoint	1
3	getthank	2
4	model	375

Top 10 Hashtags from Positive Tweets

In [21]:
```
# select top 10 hastag
d=d.nlargest(columns='Count',n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hastag',y='Count')
plt.show()
```



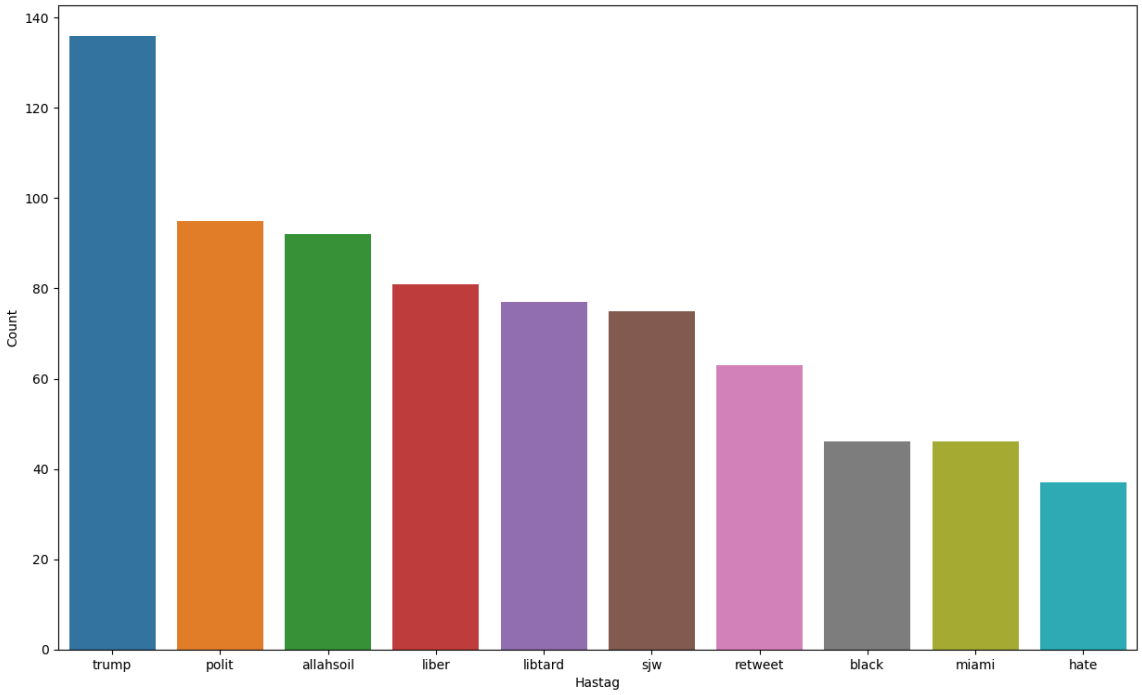
In [22]:
```
freq=nltk.FreqDist(ht_negative)
d=pd.DataFrame({'Hastag': list(freq.keys()),
                                'Count':list(freq.values())})
d.head()
```

Out[22]:

	Hastag	Count
0	cnn	10
1	michigan	2
2	tcot	14
3	australia	6
4	opkillingbay	5

Top 10 Hashtags from Negative Tweets

In [23]:
```
# select top 10 hastag
d=d.nlargest(columns='Count',n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hastag',y='Count')
plt.show()
```



In [24]:
```
df['label'].unique()
```

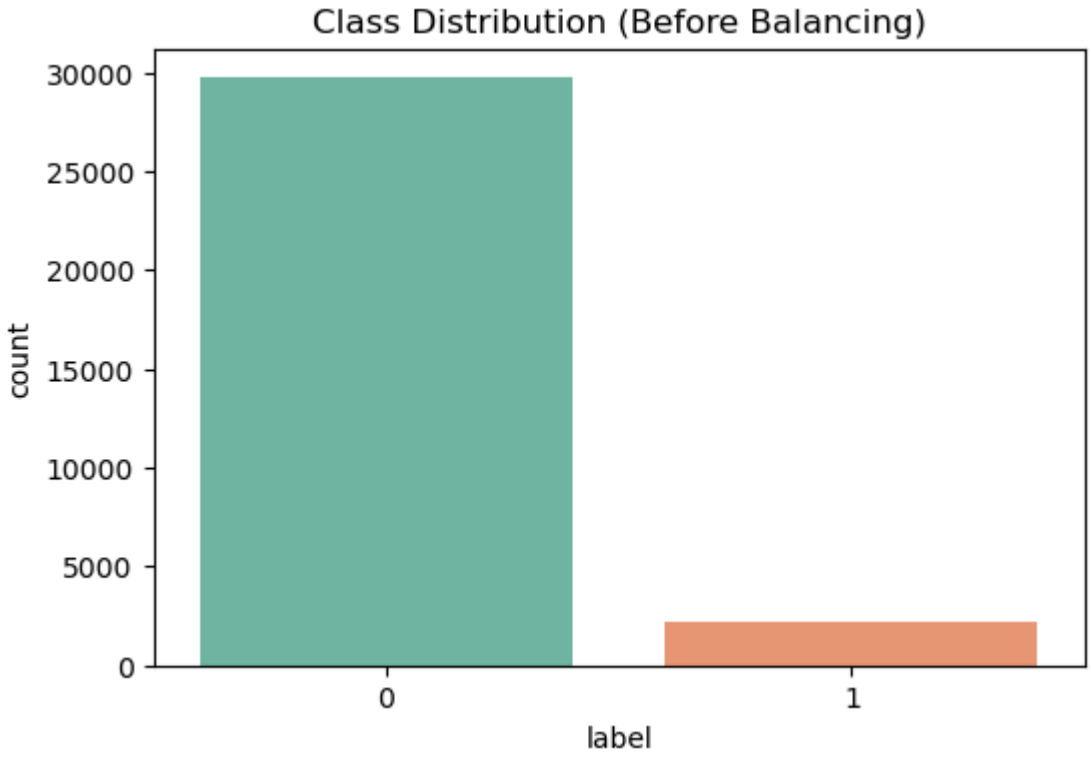
Out[24]:
```
array([0, 1], dtype=int64)
```

In [25]:
```
df['label'].value_counts()
```

Out[25]:
```
label
0    29720
1     2242
Name: count, dtype: int64
```

Visualizing Class Distribution Before Balancing

In [26]:
```
# Visualize the distribution of labels before balancing
plt.figure(figsize=(6, 4))
sns.countplot(x='label', data=df, palette='Set2')
plt.title('Class Distribution (Before Balancing)')
plt.show()
```



In [27]:
```
X = df['clean_tweet']  # Features
y = df['label']        # Target labels
```

In [28]:
```
# Feature extraction using CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, s
bow = bow_vectorizer.fit_transform(X)
```

Splitting Data into Train and Test

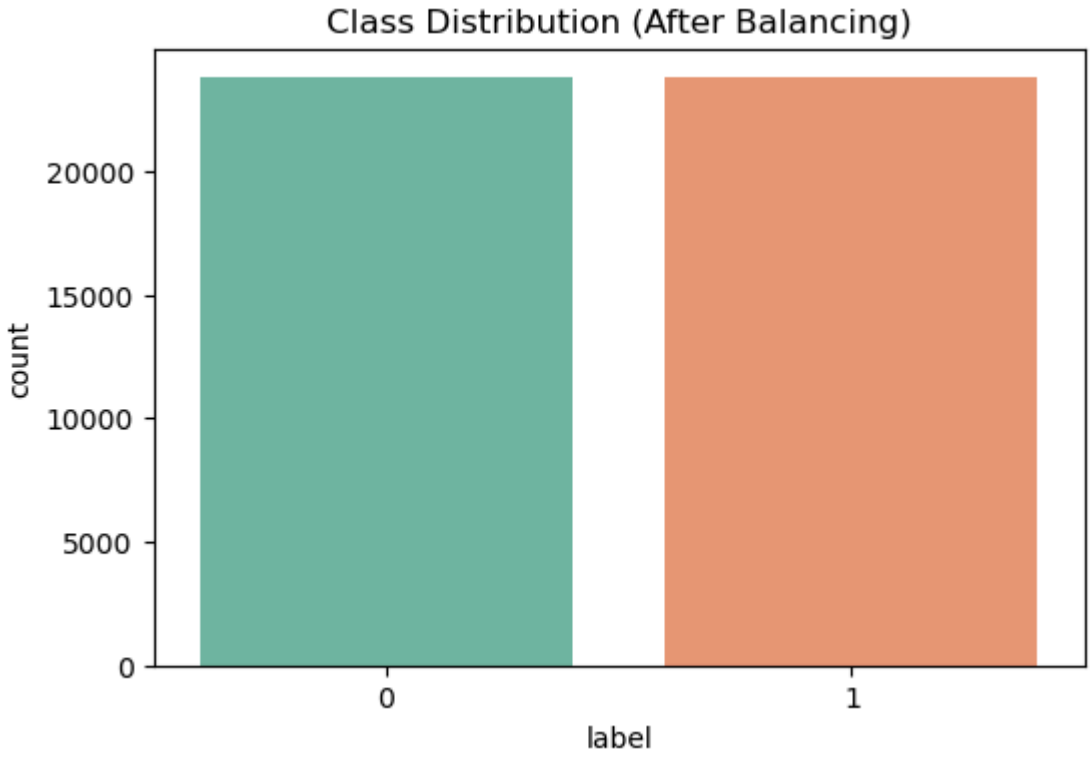
In [29]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(bow, y, random_state=42,
```

Applying SMOTE for Balancing

In [30]:
```
# Apply SMOTE to balance the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

Visualizing Class Distribution After Balancing

In [31]:
```
# Visualize the distribution of labels after balancing
plt.figure(figsize=(6, 4))
sns.countplot(x=y_train_smote, palette='Set2')
plt.title('Class Distribution (After Balancing)')
plt.show()
```



Training the Logistic Regression Model

In [32]:
```
# Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train_smote, y_train_smote)
```

Out[32]:
```
LogisticRegression
LogisticRegression()
```

Model Evaluation

In [33]:
```
# Model predictions
ypred_test= model.predict(X_test)

# Calculate F1 Score and Accuracy Score
print('f1 score:',f1_score(y_test, ypred_test))
print('accuracy score:',accuracy_score(y_test,ypred_test))
```

```
f1 score: 0.4118895966029724
accuracy score: 0.8700140778977006
```

The model achieved an accuracy of 87%, showing it correctly classified most tweets overall.

However, the F1 score of 0.412 suggests it could be better at identifying tweets from the minority class, highlighting some imbalance-related challenges

Making Predictions on New Data

In [34]:
```
# Example of new data for prediction
new_data = ["I love this product!", "This is the worst experience ever.", "
```

In [35]:
```
# Preprocess new data (apply the same cleaning steps)
new_data_clean = [remove_pattern(tweet, '@[\w]*') for tweet in new_data]
new_data_clean = [re.sub('[^a-zA-Z#]', ' ', tweet) for tweet in new_data_cle
new_data_clean = [' '.join([word for word in tweet.split() if len(word) > 3
```

In [36]:
```
# Tokenize and stem new data
new_tokenized_data = [tweet.split() for tweet in new_data_clean]
new_tokenized_data = [[stemmer.stem(word) for word in tweet] for tweet in ne
new_data_cleaned = [' '.join(tweet) for tweet in new_tokenized_data]
```

In [37]:
```
# Transform new data using the same vectorizer
new_data_bow = bow_vectorizer.transform(new_data_cleaned)
```

In [38]:
```
# Predict sentiment for the new data
new_pred = model.predict(new_data_bow)
```

In [39]:
```
# Output the predictions
print("Predictions for new data:")
for tweet, pred in zip(new_data, new_pred):
    print(f"Tweet: {tweet} | Prediction: {'Positive' if pred == 0 else 'Nega
```

```
Predictions for new data:
Tweet: I love this product! | Prediction: Positive
Tweet: This is the worst experience ever. | Prediction: Negative
Tweet: The service was amazing! | Prediction: Positive
```

In [ ]:

In [40]:
```
!pip install streamlit
```

```
Requirement already satisfied: streamlit in c:\users\user\anaconda3\lib\si
te-packages (1.39.0)
Requirement already satisfied: altair<6,>=4.0 in c:\users\user\anaconda3\l
ib\site-packages (from streamlit) (5.4.1)
Requirement already satisfied: blinker<2,>=1.0.0 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (1.8.2)
Requirement already satisfied: cachetools<6,>=4.0 in c:\users\user\anacond
a3\lib\site-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in c:\users\user\anaconda3\li
b\site-packages (from streamlit) (8.0.4)
Requirement already satisfied: numpy<3,>=1.20 in c:\users\user\anaconda3\l
ib\site-packages (from streamlit) (1.24.3)
Requirement already satisfied: packaging<25,>=20 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (23.1)
Requirement already satisfied: pandas<3,>=1.4.0 in c:\users\user\anaconda3
\lib\site-packages (from streamlit) (2.0.3)
Requirement already satisfied: pillow<11,>=7.1.0 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (9.4.0)
Requirement already satisfied: protobuf<6,>=3.20 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (4.25.3)
Requirement already satisfied: pyarrow>=7.0 in c:\users\user\anaconda3\lib
\site-packages (from streamlit) (11.0.0)
Requirement already satisfied: requests<3,>=2.27 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (2.31.0)
Requirement already satisfied: rich<14,>=10.14.0 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (13.7.1)
Requirement already satisfied: tenacity<10,>=8.1.0 in c:\users\user\anacon
da3\lib\site-packages (from streamlit) (8.2.2)
Requirement already satisfied: toml<2,>=0.10.1 in c:\users\user\anaconda3
\lib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in c:\users\use
r\anaconda3\lib\site-packages (from streamlit) (4.12.2)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\users\us
er\anaconda3\lib\site-packages (from streamlit) (3.1.43)
Requirement already satisfied: pydeck<1,>=0.8.0b4 in c:\users\user\anacond
a3\lib\site-packages (from streamlit) (0.9.1)
Requirement already satisfied: tornado<7,>=6.0.3 in c:\users\user\anaconda
3\lib\site-packages (from streamlit) (6.3.2)
Requirement already satisfied: watchdog<6,>=2.1.5 in c:\users\user\anacond
a3\lib\site-packages (from streamlit) (2.1.6)
Requirement already satisfied: jinja2 in c:\users\user\anaconda3\lib\site-
packages (from altair<6,>=4.0->streamlit) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in c:\users\user\anaconda3
\lib\site-packages (from altair<6,>=4.0->streamlit) (4.17.3)
Requirement already satisfied: narwhals>=1.5.2 in c:\users\user\anaconda3
\lib\site-packages (from altair<6,>=4.0->streamlit) (1.13.2)
Requirement already satisfied: colorama in c:\users\user\anaconda3\lib\sit
e-packages (from click<9,>=7.0->streamlit) (0.4.6)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\user\anaconda3
\lib\site-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.11)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\ana
conda3\lib\site-packages (from pandas<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\anaconda3\lib
\site-packages (from pandas<3,>=1.4.0->streamlit) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\user\anaconda3\l
ib\site-packages (from pandas<3,>=1.4.0->streamlit) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\user\a
naconda3\lib\site-packages (from requests<3,>=2.27->streamlit) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\anaconda3\lib
\site-packages (from requests<3,>=2.27->streamlit) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\user\anacond
```

```
a3\lib\site-packages (from requests<3,>=2.27->streamlit) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\anacond
a3\lib\site-packages (from requests<3,>=2.27->streamlit) (2023.7.22)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\user\anac
onda3\lib\site-packages (from rich<14,>=10.14.0->streamlit) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\user\an
aconda3\lib\site-packages (from rich<14,>=10.14.0->streamlit) (2.15.1)
Requirement already satisfied: smmap<6,>=3.0.1 in c:\users\user\anaconda3
\lib\site-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->st
reamlit) (5.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\user\anaconda3
\lib\site-packages (from jinja2->altair<6,>=4.0->streamlit) (2.1.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\user\anaconda3\li
b\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.1
4.0 in c:\users\user\anaconda3\lib\site-packages (from jsonschema>=3.0->al
tair<6,>=4.0->streamlit) (0.18.0)
Requirement already satisfied: mdurl~=0.1 in c:\users\user\anaconda3\lib\s
ite-packages (from markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit)
(0.1.0)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\sit
e-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.1
6.0)


[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Saving the Trained Model and Vectorizer

In [41]:
```
import joblib
joblib.dump(model, 'logistic_model.pkl')
joblib.dump(bow_vectorizer, 'bow_vectorizer.pkl')
```

Out[41]:
```
['bow_vectorizer.pkl']
```

Import Required Libraries for deployment

In [42]:
```
import streamlit as st
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
import nltk
from nltk.stem.porter import PorterStemmer
import joblib
```

Load Pre-trained Model and Vectorizer

In [43]:
```
# Load pre-trained model and vectorizer
model = joblib.load('logistic_model.pkl')
vectorizer = joblib.load('bow_vectorizer.pkl')
stemmer = PorterStemmer()
```

Define Text Preprocessing Function

In [44]:
```
# Define text preprocessing function
def preprocess_text(text):
    text = re.sub('@[\w]*', '', text)  # Remove @mentions
    text = re.sub('[^a-zA-Z#]', ' ', text)  # Remove special characters and
    text = ' '.join([word for word in text.split() if len(word) > 3])  # Re
    text = ' '.join([stemmer.stem(word) for word in text.split()])  # Stemm
    return text
```

Build the Streamlit Interface

In [45]:
```
# Streamlit UI
st.title("Twitter Sentiment Analysis")
user_input = st.text_area("Enter a tweet for sentiment prediction:")
```

```
2024-11-12 13:52:51.528 WARNING streamlit.runtime.scriptrunner_utils.scrip
t_run_context: Thread 'MainThread': missing ScriptRunContext! This warning
can be ignored when running in bare mode.
2024-11-12 13:52:53.665
  Warning: to view this Streamlit app on a browser, run it with the follow
ing
  command:

    streamlit run C:\Users\user\anaconda3\Lib\site-packages\ipykernel_laun
cher.py [ARGUMENTS]
2024-11-12 13:52:53.669 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.673 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.677 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.682 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.685 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.689 Session state does not function when running a scr
ipt without `streamlit run`
2024-11-12 13:52:53.692 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.695 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
```

Predict Sentiment of the User's Input

In [46]:
```
if st.button("Predict Sentiment"):
    # Preprocess the input
    clean_input = preprocess_text(user_input)
    # Vectorize the input text
    input_vector = vectorizer.transform([clean_input])
    # Make prediction
    prediction = model.predict(input_vector)[0]
    sentiment = "Positive" if prediction == 0 else "Negative"
    st.write(f"Sentiment: {sentiment}")
```

```
2024-11-12 13:52:53.734 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.745 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.746 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.748 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
2024-11-12 13:52:53.750 Thread 'MainThread': missing ScriptRunContext! Thi
s warning can be ignored when running in bare mode.
```

Saving the Streamlit Code to app.py

In [47]:
```
code = """
import streamlit as st
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
import nltk
from nltk.stem.porter import PorterStemmer
import joblib

# Load pre-trained model and vectorizer
model = joblib.load('logistic_model.pkl')
vectorizer = joblib.load('bow_vectorizer.pkl')
stemmer = PorterStemmer()

# Define text preprocessing function
def preprocess_text(text):
    text = re.sub('@[\\w]*', '', text)  # Remove @mentions
    text = re.sub('[^a-zA-Z#]', ' ', text)  # Remove special characters and
    text = ' '.join([word for word in text.split() if len(word) > 3])  # Rem
    text = ' '.join([stemmer.stem(word) for word in text.split()])  # Stemmi
    return text

# Streamlit UI
st.title("Twitter Sentiment Analysis")
user_input = st.text_area("Enter a tweet for sentiment prediction:")

if st.button("Predict Sentiment"):
    # Preprocess the input
    clean_input = preprocess_text(user_input)
    # Vectorize the input text
    input_vector = vectorizer.transform([clean_input])
    # Make prediction
    prediction = model.predict(input_vector)[0]
    sentiment = "Positive" if prediction == 0 else "Negative"
    st.write(f"Sentiment: {sentiment}")
"""
```

In [48]:
```
# Save the code to 'app.py'
with open("app.py", "w") as f:
    f.write(code)

print("Code has been saved as 'app.py'.")
```

```
Code has been saved as 'app.py'.
```

In [49]:
```
cd path\to\your\directory
```

```
[WinError 3] The system cannot find the path specified: 'path\\to\\your\\d
irectory'
D:\Data Science
```

In [50]:
```
cd D:\Data Science
```

```
D:\Data Science
```

In [51]:
```
dir
```

Out[51]:
```
<function dir>
```

Running the Streamlit App Programmatically

In [*]:
```
import subprocess

subprocess.run(["streamlit", "run", "app.py"])
```

In [ ]:

In [ ]:

In [ ]:

Project Summary: Twitter Sentiment Analysis Web Application

This project focuses on building a Twitter Sentiment Analysis application that classifies tweets as either positive or negative using Logistic Regression and Streamlit for deployment. The key steps involved in the project are as follows:

1.Data Preprocessing: Tweets are cleaned by removing unwanted elements such as @mentions, special characters, and short words. Text stemming is applied to ensure that words are reduced to their root form.

2.Feature Extraction: A CountVectorizer is used to convert text into a numeric form (bag-of-words), which is used to train the sentiment analysis model.

3.Model Training: A Logistic Regression model is trained on preprocessed tweet data to predict the sentiment of new tweets.

4.Model Serialization: The trained model and the feature vectorizer are serialized using joblib for easy reuse in the deployment phase

5.Streamlit Application:

The application allows users to enter a tweet, which is processed and classified as positive or negative. The results are displayed in real-time via the Streamlit interface.

6.Automation:

The subprocess module is used to programmatically run the Streamlit app, simplifying the process of starting the web application.

In [ ]: