Sentiment Analysis of IMDB Movie Reviews

Submitted By :

Beeram Sai Sarvagna (AM.EN.U4CSE20316)

Krishnapriya Dinesan (AM.EN.U4CSE20339)

Nayan Thara M (AM.EN.U4CSE20347)

T S N Manikanta (AM.EN.U4CSE20370)

Yeganathan S (AM.EN.U4CSE20376)

DATASET1

Import Necessary Packages

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import spacy
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from textblob import TextBlob
from textblob import Word
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Load Dataset

In [2]:
```
df = pd.read_csv('IMDB Dataset.csv')

df.head()
```

Out[2]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Inference: Here we are loading the dataset and using head() we can print the first 5 rows of the data set.

Data Preprocessing :

In [3]:
```
df.dropna(axis=1)
```

Out[3]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns

Inference: Here we are dropping the Null value Columns if they are present.

In [4]:
```
#Removing null values
df.dropna(inplace=True)
```

Inference: Here we are dropping the Null value Columns if they are present.

- Cleaning Dataset

```
In [5]: #Function to remove all unwanted characters in our data
def clean(text):
    text = re.sub(r'http://S+|https://S+',' ',text)
    text = re.sub("[^a-zA-Z]+"," ",text)
    text = re.sub(" s "," ",text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r"rt ", "", text)
    text = text.lower()
    text = text.lstrip().rstrip()
    return text
```

```
In [6]: cleaned_df = [clean(text) for text in df["review"]]
```

Inference: Here we are cleaning the data if it contains some special characters, Https Links, checking for only alphabets, removing the extra spaces and all.

```
In [7]: print(cleaned_df[1:100])
```

```
['a wonderful little production br br the filming technique is very unassuming very old time bbc fashion and gives a comforti
ng and sometimes discomforting sense of realism to the entire piece br br the actors are extremely well chosen michael sheen
not only has got all the polari but he has all the voices down pat too you can truly see the seamless editing guided by the r
eferences to williams diary entries not only is it well worth the watching but it is a terrificly written and performed piece
a masterful production about one of the great master of comedy and his life br br the realism really comes home with the litt
le things the fantasy of the guard which rather than use the traditional dream techniques remains solid then disappears it pl
ays on our knowledge and our senses particularly with the scenes concerning orton and halliwell and the sets particularly of
their flat with halliwell murals decorating every surface are terribly well done', 'i thought this was a wonderful way to spe
nd time on a too hot summer weekend sitting in the air conditioned theater and watching a light hearted comedy the plot is si
mplistic but the dialogue is witty and the characters are likable even the well bread suspected serial killer while some may
be disappointed when they realize this is not match point risk addiction i thought it was proof that woody allen is still ful
ly in control of the style many of us have grown to love br br this was the most i d laughed at one of woody comedies in year
s dare i say a decade while i ve never been impressed with scarlet johanson in this she managed to tone down her sexy image a
nd jumped right into a average but spirited young woman br br this may not be the crown jewel of his career but it was wittie
r than devil wears prada and more interesting than superman a great comedy to go see with friends', 'basically there a family
where a little boy jake thinks there a zombie in his closet his parents are fighting all the time br br this movie is slower
than a soap opera and suddenly jake decides to become rambo and kill the zombie br br ok first of all when you re going to ma
ke a film you must decide if its a thriller or a drama as a drama the movie is watchable parents are divorcing arguing like i
n real life and then we have jake with his closet which totally ruins all the film i expected to see a boogeyman similar movi
```

Inference: A short example of the cleaned data by using the Function.

- Tokenization

Inference: Tokenization can separate sentences, words, characters, or subwords. When we split the text into sentences. Here we are separating the sentences which we already cleaned and sent to a cleaned_df list.

```
In [8]: x = [nltk.word_tokenize(w) for w in cleaned_df]
```

- Lemmatization

```
In [9]: lemma = WordNetLemmatizer()
lemmatized = [[lemma.lemmatize(w) for w in text] for text in x]
```

Inference: Lemmatization is the process of grouping together the different inflected forms of a word, so they can be analysed as a single item. So we link words with similar meanings to one word. For example, the word better can be changed to good.Here we are lemmatizing the list that contains the tokenized words.

- Removal of Stopwords

```
In [10]: without_stopwords = [[w for w in text if w not in stopwords.words('english')] for text in lemmatized]
```

Inference: Stop words are a set of commonly used words in a language. They are used to eliminating words that are so commonly used that they carry very little useful information. Here we use stop words that are in English such as “a”, “the”, “is”, “are” etc.After removing the stop words we are storing those to a variable without_stopwords

- Vectorization

```
In [11]: vectorizer = CountVectorizer(max_features=20000,analyzer='word',stop_words = 'english')
         X = vectorizer.fit_transform([' '.join(text) for text in without_stopwords]).toarray()
```

```
In [29]: X
```

```
Out[29]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Inference: Vectorization is a step in feature extraction. The idea is to get some distinct features out of the text for the model to train on, by converting text to numerical vectors. The vectorized data are then converted to array and stored in variable x.

Data Summarization :

```
In [12]: df.describe()
```

Out[12]:

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

Inference: describes gives us the count,unique values and frequency in a dataset.

```
In [13]: df['sentiment'].value_counts()
```

```
Out[13]: positive    25000
         negative    25000
         Name: sentiment, dtype: int64
```

Inference: we are getting the Frequency of each word.

```
In [14]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 1.1+ MB
```

Inference: info we are using to get a summary of the dataframe.

```
In [15]: print(df.isnull().sum())
```

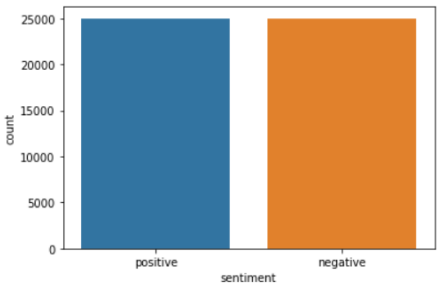
```
review       0
sentiment    0
dtype: int64
```

Inference: To check the number of null values present in each column.

Data Visualization :

- Show the counts of observations in each categorical bin using bars

```
In [16]: sns.countplot(x='sentiment',data = df)
         plt.show()
```

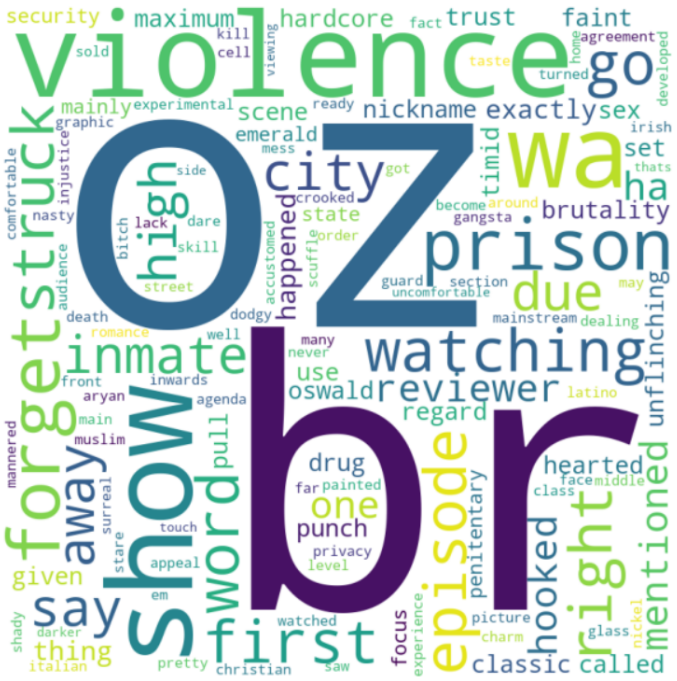


Inference: Here we have used countplot() to show the counts of observations in each categorical bin using bars.

- Show words that doesn't include stop words in tight layout of words

```
In [17]: w = [' '.join(text) for text in without_stopwords]
         wordcloud = WordCloud(width = 800, height = 800,
                               background_color ='white',
                               min_font_size = 10).generate(w[0])

         plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)
         plt.show()
```

Output:



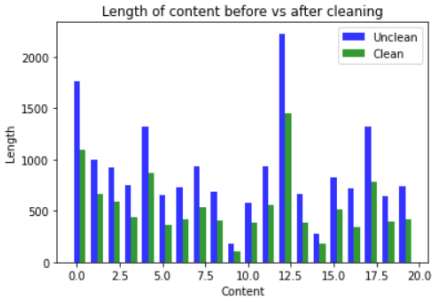
Inference: Word Cloud is a data visualisation technique used for representing text data in which the size of each word indicates its frequency or importance.

- Show clean and unclean data in the bar graph from dataset

In [18]:
```
unclean = [len(x) for x in df["review"]]
clean = [len(y) for y in w]

n_groups = 20
a = unclean[:20]
b = clean[:20]

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, a, bar_width,
                 alpha=opacity,
                 color='b',
                 label='Unclean')

rects2 = plt.bar(index + bar_width, b, bar_width,
                 alpha=opacity,
                 color='g',
                 label='Clean')

plt.xlabel('Content')
plt.ylabel('Length')
plt.title('Length of content before vs after cleaning')
plt.legend()
```

Output:

Out[18]:
```
<matplotlib.legend.Legend at 0x1da008ec2b0>
```

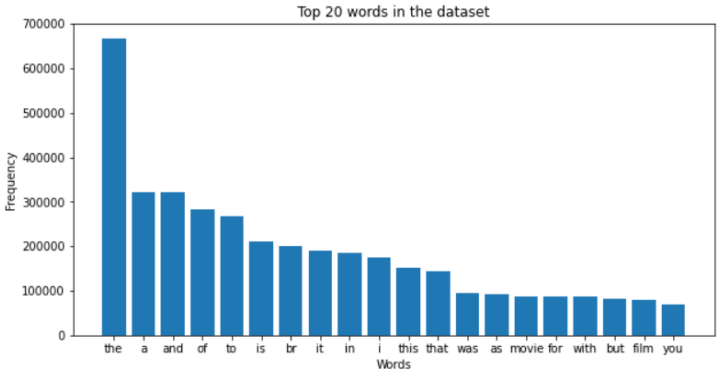


Inference: The following graph depicts the comparison between unclean and cleaned data length of content.

- Show top 20 words with the highest frequency in the plot

In [18]:
```
def top_words(cleaned_df):
    # create a dictionary to store the words and their frequency
    word_dict = {}
    for text in cleaned_df:
        for word in text.split():
            if word in word_dict:
                word_dict[word] += 1
            else:
                word_dict[word] = 1
    # sort the dictionary by values
    sorted_dict = sorted(word_dict.items(), key=lambda x: x[1], reverse=True)
    # get the top 20 words
    top_words = sorted_dict[:20]
    # get the top 20 words and their frequency
    top_words, top_freq = zip(*top_words)
    # plot the top 20 words and their frequency
    plt.figure(figsize=(10, 5))
    plt.bar(top_words, top_freq)
    plt.title('Top 20 words in the dataset')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.show()

top_words(cleaned_df)
```

Output:



Inference: The above graph shows us the top 20 words according to their frequencies in decreasing order.

Cleaned Data Saving

In [27]:
```
# save list to csv file
def save_list(without_stopwords, cleaneddataset1):
    # convert list to dataframe
    df = pd.DataFrame(without_stopwords)
    # save to csv file
    df.to_csv(cleaneddataset1, index= False)
```

In [28]:
```
save_list(w, 'cleaneddataset1.csv')
```

Inference: Here we are saving the Cleaned data into a new csv File.

Original Dataset: CSV FILE

Cleaned Dataset: CSV FILE

DATASET2

Import Required Packages

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import spacy
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from textblob import TextBlob
from textblob import Word
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Loading Dataset

In [2]:
```
df = pd.read_csv('movie review.csv',sep='\t')

df.head()
```

Out[2]:

	id	review	sentiment
0	5814_8	With all this stuff going down at the moment w...	1
1	2381_9	\The Classic War of the Worlds\" by Timothy Hi...	1
2	7759_3	The film starts with a manager (Nicholas Bell)...	0
3	3630_4	It must be assumed that those who praised this...	0
4	9495_8	Superbly trashy and wondrously unpretentious 8...	1

Inference: Here we are loading the dataset and using head() we can print the first 5 rows of the data set.

In [3]:
```
#Dropping unwanted columns
df.drop('id',inplace=True,axis=1)
```

In [4]:
```
df.head()
```

Out[4]:

	review	sentiment
0	With all this stuff going down at the moment w...	1
1	\The Classic War of the Worlds\" by Timothy Hi...	1
2	The film starts with a manager (Nicholas Bell)...	0
3	It must be assumed that those who praised this...	0
4	Superbly trashy and wondrously unpretentious 8...	1

Inference: Removed the Unnecessary Column id from the dataframe.

Data Preprocessing :

In [5]:
```
df.dropna(axis=1)
```

Out[5]:

	review	sentiment
0	With all this stuff going down at the moment w...	1
1	\The Classic War of the Worlds\" by Timothy Hi...	1
2	The film starts with a manager (Nicholas Bell)...	0
3	It must be assumed that those who praised this...	0
4	Superbly trashy and wondrously unpretentious 8...	1
...	...	...
22495	It seems like more consideration has gone into...	0
22496	I don't believe they made this film. Completel...	0
22497	Guy is a loser. Can't get girls, needs to buil...	0
22498	This 30 minute documentary BuÃ±uel made in the...	0
22499	I saw this movie as a child and it broke my he...	1

22500 rows × 2 columns

Inference: Here we are dropping the Null value Columns if they are present.

In [6]:
```
#Removing null values
df.dropna(inplace=True)
```

Inference: As we saw earlier, rather than imputing missing values we are dropping the columns in the dataframe that have the missing values.

In [7]:
```
#Function to remove all unwanted characters in our data
def clean(text):
    text = re.sub(r'http://S+|https://S+',' ',text)
    text = re.sub("[^a-zA-Z]+"," ",text)
    text = re.sub(" s "," ",text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r"rt ", "", text)
    text = text.lower()
    text = text.lstrip().rstrip()
    return text
```

In [8]:
```
cleaned_df = [clean(text) for text in df["review"]]
```

Inference: Cleaning the data if it contains some special characters , Https Links, checking for only alphabets, removing the extra spaces and also to remove duplicate categorization from your data list and streamline the data so it becomes error-free.

In [9]:
```
print(cleaned_df[1:100])
```

```
re most fearsome predators the sabretooth tiger or smilodon scientific ambition turns deadly however and when the high voltag
e fence is opened the creature escape and begins savagely stalking its prey the human visitors tourists and scientific meanwh
ile some youngsters enter in the restricted area of the security center and are attacked by a pack of large pre historical an
imals which are deadlier and bigger in addition a security agent stacy haiduk and her mate brian wimmer fight hardly against
the carnivorous smilodons the sabretooths themselves of course are the real star stars and they are astounding terrifyingly t
hough not convincing the giant animals savagely are stalking its prey and the group run afoul and fight against one nature mo
st fearsome predators furthermore a third sabretooth more dangerous and slow stalks its victims br br the movie delivers the
goods with lots of blood and gore as beheading hair raising chills full of scares when the sabretooths appear with mediocre s
pecial effects the story provides exciting and stirring entertainment but it results to be quite boring the giant animals are
majority made by computer generator and seem totally lousy middling performances though the players reacting appropriately to
becoming food actors give vigorously physical performances dodging the beasts running bound and leaps or dangling over walls
and it packs a ridiculous final deadly scene no for small kids by realistic gory and violent attack scenes other films about
sabretooths or smilodon are the following sabretooth by james r hickox with vanessa angel david keith and john rhys davies an
d the much better bc by roland emmerich with with steven strait cliff curtis and camilla belle this motion picture filled wit
h bloody moments is badly directed by george miller and with no originality because takes too many elements from previous fil
ms miller is an australian director usually working for television tidal wave journey to the center of the earth and many oth
ers and occasionally for cinema the man from snowy river zeus and roxanne robinson crusoe rating below average bottom of barr
el', 'it must be assumed that those who praised this film the greatest filmed opera ever didn t i read somewhere either don t
care for opera don t care for wagner or don t care about anything except their desire to appear cultured either as a represen
tation of wagner swan song or as a movie this strikes me as an unmitigated disaster with a leaden reading of the score matche
```

Inference: Here is the cleaned data by using the Function.

- Tokenization

In [10]:
```
x = [nltk.word_tokenize(w) for w in cleaned_df]
```

Inference: As mentioned above we are dividing the texts into words or smaller sub-texts, which will enable good generalisation of relationship between the texts and the labels and then we send this data to a cleaned_df list.

- Lemmatization

In [12]:
```
lemma = WordNetLemmatizer()
lemmatized = [[lemma.lemmatize(w) for w in text] for text in x]
```

Inference: Lemmatization is the process of grouping together the different inflected forms of a word, so they can be analysed as a single item. So we link words with similar meanings to one word. For example, the word better can be changed to good.
Here we are lemmatizing the list that contains the tokenized words.

- Removal of Stopwords

```
In [13]: without_stopwords = [[w for w in text if w not in stopwords.words('english')] for text in lemmatized]
```

Inference: Stop words are a set of commonly used words in a language. They are used to eliminating words that are so commonly used that they carry very little useful information. Here we use stop words that are in English such as “a”, “the”, “is”, “are” etc.
After removing the stop words we are storing those to a variable without_stopwords

- Vectorization

```
In [30]: vectorizer = CountVectorizer(max_features=50000,analyzer='word',stop_words = 'english')
         X = vectorizer.fit_transform([' '.join(text) for text in without_stopwords]).toarray()
```

Inference: Vectorization is a step in feature extraction. The idea is to get some distinct features out of the text for the model to train on, by converting text to numerical vectors. The vectorized data are then converted to array and stored in variable x.

Data Summarization :

```
In [14]: df.describe()
```

Out[14]:

	sentiment
count	22500.000000
mean	0.501244
std	0.500010
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

Inference: “describes” gives us the count, unique values and frequency in a dataset.

```
In [15]: df['sentiment'].value_counts()
Out[15]: 1    11278
         0    11222
         Name: sentiment, dtype: int64
```

Inference: “value_counts()” returns us object containing counts of unique values
Here 11278 unique values in 1st column and 11222 unique values in 0th column.

```
In [16]: df.info()
         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 22500 entries, 0 to 22499
         Data columns (total 2 columns):
          #   Column     Non-Null Count  Dtype
         ---  ------     --------------  -----
          0   review     22500 non-null  object
          1   sentiment  22500 non-null  int64
         dtypes: int64(1), object(1)
         memory usage: 527.3+ KB
```

Inference: “info” we are using to get a summary of the dataframe.

```
In [17]: print(df.isnull().sum())
```

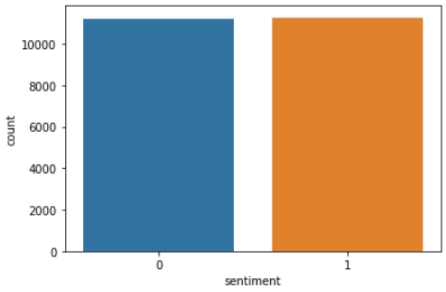
```
review       0
sentiment    0
dtype: int64
```

Inference: Counting the number of missing values in each row and column by calling sum() from the result of isnull().

Data Visualization :

- Show the counts of observations in each categorical bin using bars

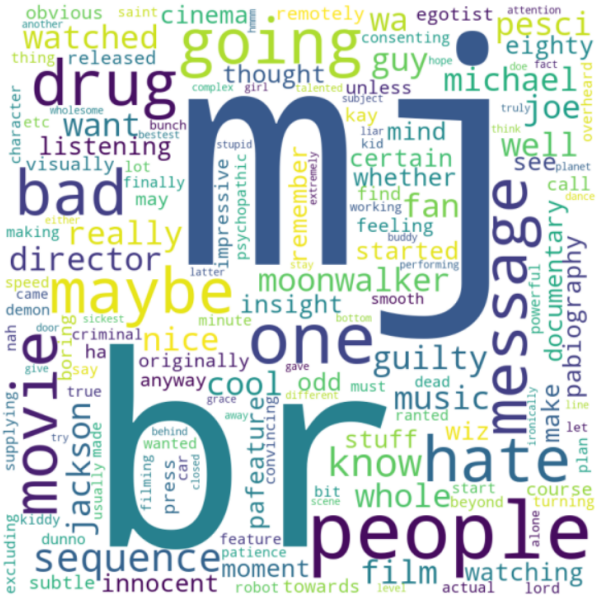
```
In [18]: sns.countplot(x='sentiment',data = df)
plt.show()
```



Inference: Using countplot() to show the counts of observations in each categorical bin using bars as a histogram across a categorical data.

- Show words that doesn't include stop words in tight layout of words

```
In [19]: w = [' '.join(text) for text in without_stopwords]
wordcloud = WordCloud(width = 800, height = 800,
                      background_color ='white',
                      min_font_size = 10).generate(w[0])

plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

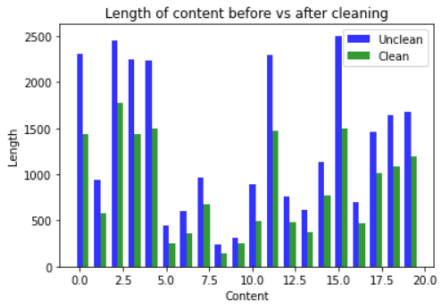


Inference: Word Cloud is a data visualisation technique used for representing text data in which the size of each word indicates its frequency or importance.

- Show clean and unclean data in the bar graph from dataset

In [20]:
```
unclean = [len(x) for x in df["review"]]
clean = [len(y) for y in w]

n_groups = 20
a = unclean[:20]
b = clean[:20]

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, a, bar_width,
                 alpha=opacity,
                 color='b',
                 label='Unclean')

rects2 = plt.bar(index + bar_width, b, bar_width,
                 alpha=opacity,
                 color='g',
                 label='Clean')

plt.xlabel('Content')
plt.ylabel('Length')
plt.title('Length of content before vs after cleaning')
plt.legend()
```

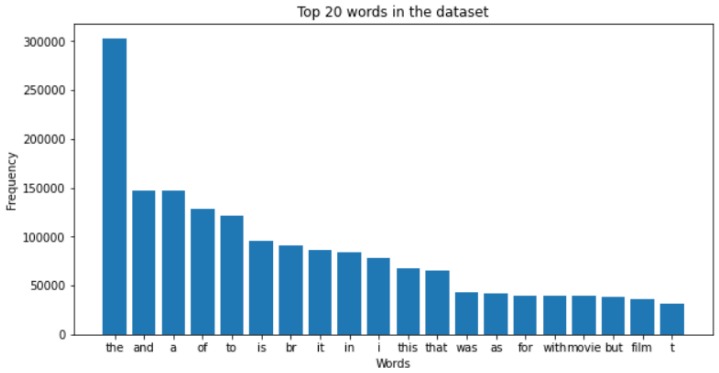
Out[20]:
```
<matplotlib.legend.Legend at 0x254df7a7430>
```



Inference: The following graph depicts the comparison between unclean and cleaned data length of content.

- Show top 20 words with the highest frequency in the plot

In [13]:
```
def top_words(cleaned_df):
    # create a dictionary to store the words and their frequency
    word_dict = {}
    for text in cleaned_df:
        for word in text.split():
            if word in word_dict:
                word_dict[word] += 1
            else:
                word_dict[word] = 1
    # sort the dictionary by values
    sorted_dict = sorted(word_dict.items(), key=lambda x: x[1], reverse=True)
    # get the top 20 words
    top_words = sorted_dict[:20]
    # get the top 20 words and their frequency
    top_words, top_freq = zip(*top_words)
    # plot the top 20 words and their frequency
    plt.figure(figsize=(10, 5))
    plt.bar(top_words, top_freq)
    plt.title('Top 20 words in the dataset')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.show()

top_words(cleaned_df)
```



Inference: We are iterating over the list and use each distinct element of the list as a key of the dictionary and store the corresponding count of that key as values. The above graph shows us the top 20 words according to their frequencies in decreasing order.

Cleaned dataset

In [25]:
```
# save list to csv file
def save_list(without_stopwords, cleaneddataset1):
    # convert list to dataframe
    df = pd.DataFrame(without_stopwords)
    # save to csv file
    df.to_csv(cleaneddataset1, index= False)
```

In [26]:
```
save_list(w, 'cleaneddataset2.csv')
```

Inference: Exporting a Pandas DataFrame to a CSV file.
Pandas enables us to do so with its inbuilt to_csv() function then, we record the cleaned data to a new csv file.

Original Dataset: CSV FILE

Cleaned Dataset: CSV FILE

DATASET 3

Import Required Packages

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import spacy
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from textblob import TextBlob
from textblob import Word
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Load Dataset

```
In [20]: df = pd.read_csv('dataset3.csv')

         df.head()
```

Out[20]:

	text	label
0	I grew up (b. 1965) watching and loving the Th...	0
1	When I put this movie in my DVD player, and sa...	0
2	Why do people who do not know what a particula...	0
3	Even though I have great interest in Biblical ...	0
4	Im a die hard Dads Army fan and nothing will e...	1

Checking data set for null values

```
In [3]: df.dropna(axis=1)
```

Out[3]:

	text	label
0	I grew up (b. 1965) watching and loving the Th...	0
1	When I put this movie in my DVD player, and sa...	0
2	Why do people who do not know what a particula...	0
3	Even though I have great interest in Biblical ...	0
4	Im a die hard Dads Army fan and nothing will e...	1
...	...	...
39995	"Western Union" is something of a forgotten cl...	1
39996	This movie is an incredible piece of work. It ...	1
39997	My wife and I watched this movie because we pl...	0
39998	When I first watched Flatliners, I was amazed....	1
39999	Why would this film be so good, but only gross...	1

40000 rows × 2 columns

Inference: cleaned the data set by removing any null values.

Data Preprocessing :

- Checking data set for null values

```
In [4]: #Removing null values
        df.dropna(inplace=True)
```

Inference: cleaned the data set by removing any null values.

- Optimising data by removing unwanted characters

```
In [5]: #Function to remove all unwanted characters in our data
        def clean(text):
            text = re.sub(r'http://S+|https://S+',' ',text)
            text = re.sub("[^a-zA-Z]+"," ",text)
            text = re.sub(" s "," ",text)
            text = re.sub(r'[^\w\s]', '', text)
            text = re.sub(r"rt ", "", text)
            text = text.lower()
            text = text.lstrip().rstrip()
            return text
```

```
In [6]: cleaned_df = [clean(text) for text in df["text"]]
```

Inference: Here we are cleaning the data if it contains some special characters , Https Links, checking for only alphabets, removing the extra spaces and all.

Output:

```
In [7]: print(cleaned_df[1:100])
```

```
['when i put this movie in my dvd player and sat down with a coke and some chips i had some expectations i was hoping that th
is movie would contain some of the strong points of the first movie awsome animation good flowing story excellent voice cast
funny comedy and a kick ass soundtrack but to my disappointment not any of this is to be found in atlantis milo return had i
read some reviews first i might not have been so let down the following paragraph will be directed to those who have seen the
first movie and who enjoyed it primarily for the points mentioned br br when the first scene appears your in for a shock if y
ou just picked atlantis milo return from the display case at your local videoshop or whatever and had the expectations i had
the music feels as a bad imitation of the first movie and the voice cast has been replaced by a not so fitting one with the e
xception of a few characters like the voice of sweet the actual drawings isnt that bad but the animation in particular is a s
ad sight the storyline is also pretty weak as its more like three episodes of schooby doo than the single adventurous story w
e got the last time but dont misunderstand it not very good schooby doo episodes i didnt laugh a single time although i might
have sniggered once or twice br br to the audience who haven t seen the first movie or don t especially care for a similar se
quel here is a fast review of this movie as a stand alone product if you liked schooby doo you might like this movie if you d
idn t you could still enjoy this movie if you have nothing else to do and i suspect it might be a good kids movie but i would
n t know it might have been better if milo return had been a three episode series on a cartoon channel or on breakfast tv',
'why do people who do not know what a particular time in the past was like feel the need to try to define that time for other
s replace woodstock with the civil war and the apollo moon landing with the titanic sinking and you ve got as realistic a fli
ck as this formulaic soap opera populated entirely by low life trash is this what kids who were too young to be allowed to go
to woodstock and who failed grade school composition do i ll show those old meanies i ll put out my own movie and prove that
you don t have to know nuttin about your topic to still make money yeah we already know that the one thing watching this film
```

- Tokenization

```
In [8]: x = [nltk.word_tokenize(w) for w in cleaned_df]
```

Inference: Tokenization can separate sentences, words, characters, or subwords. When we split the text into sentences. Here we are separating the sentences which we already cleaned and are sending to a cleaned_df list.

- Lemmatization

```
In [9]: lemma = WordNetLemmatizer()
        lemmatized = [[lemma.lemmatize(w) for w in text] for text in x]
```

Inference: Lemmatization is the process of grouping together the different inflected forms of a word, so they can be analysed as a single item. So we link words with similar meanings to one word. For example, the word better can be changed to good.
Here we are lemmatizing the list that contains the tokenized words and is storing it to a variable lemmatized.

- Removal of Stop words

```
In [10]: without_stopwords = [[w for w in text if w not in stopwords.words('english')] for text in lemmatized]
```

Inference: Stop words are a set of commonly used words in a language. They are used to eliminating words that are so commonly used that they carry very little useful information. Here we use stop words that are in English such as “a”, “the”, “is”, “are” etc.
After removing the stop words we are storing those to a variable without_stopwords

- Vectorization

```
In [42]: vectorizer = CountVectorizer(max_features=20000,analyzer='word',stop_words = 'english')
         X = vectorizer.fit_transform([' '.join(text) for text in without_stopwords]).toarray()
```

Inference: Vectorization is a step in feature extraction. The idea is to get some distinct features out of the text for the model to train on, by converting text to numerical vectors. The vectorized data are then converted to array and stored in variable x.

Data Summarization :

- Details of the data within the database.

```
In [12]: df.describe()
```

Out[12]:

	label
count	40000.000000
mean	0.499525
std	0.500006
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

Inference:

- count - The number of not-empty values.
- mean - The average (mean) value.
- std - The standard deviation.
- min - the minimum value.
- 25% - The 25% percentile.
- 50% - The 50% percentile.
- 75% - The 75% percentile.
- max - the maximum value.
  Percentile means how many of the values are less than the given percentile.

- Frequency of dataset

```
In [13]: df['label'].value_counts()
```

```
Out[13]: 0    20019
         1    19981
         Name: label, dtype: int64
```

Inference: 0 stands for positive and 1 stands for negative reviews

- Information about the data set

```
In [14]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40000 entries, 0 to 39999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    40000 non-null  object
 1   label   40000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 937.5+ KB
```

```
In [15]: print(df.isnull().sum())
```

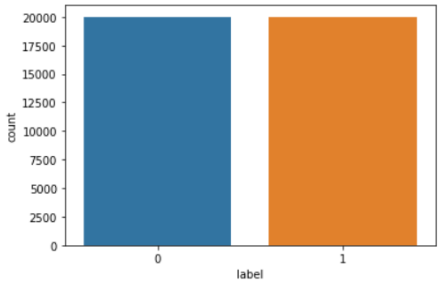
```
text     0
label    0
dtype: int64
```

Inference: There are total 40000 rows with no null values

Data Visualization :

- Show the counts of observations in each categorical bin using bars

```
In [16]: sns.countplot(x='label',data = df)
         plt.show()
```



Inference: We have equal no of positive and negative review in the sample

- Show words that doesn't include stop words in tight layout of words

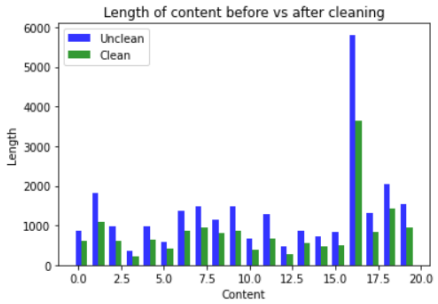
```
In [17]: w = [' '.join(text) for text in without_stopwords]
         wordcloud = WordCloud(width = 800, height = 800,
                               background_color ='white',
                               min_font_size = 10).generate(w[0])

         plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)
         plt.show()
```

Inference: Displays some sample words in the dataset other than the stopwords.

- Show clean and unclean data in the bar graph from dataset

In [18]:
```
unclean = [len(x) for x in df["text"]]
clean = [len(y) for y in w]

n_groups = 20
a = unclean[:20]
b = clean[:20]

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, a, bar_width,
                 alpha=opacity,
                 color='b',
                 label='Unclean')

rects2 = plt.bar(index + bar_width, b, bar_width,
                 alpha=opacity,
                 color='g',
                 label='Clean')

plt.xlabel('Content')
plt.ylabel('Length')
plt.title('Length of content before vs after cleaning')
plt.legend()
```

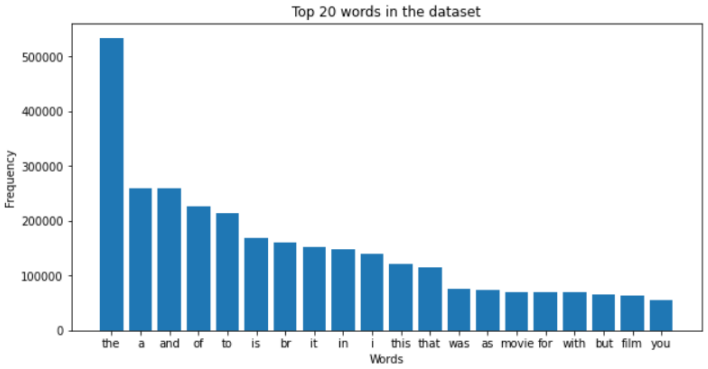
Out[18]:
```
<matplotlib.legend.Legend at 0x2399a222670>
```



Inference: Compares the amount of data before and after cleaning.

- Show top 20 words with the highest frequency in the plot

In [8]:
```
def top_words(cleaned_df):
    # create a dictionary to store the words and their frequency
    word_dict = {}
    for text in cleaned_df:
        for word in text.split():
            if word in word_dict:
                word_dict[word] += 1
            else:
                word_dict[word] = 1
    # sort the dictionary by values
    sorted_dict = sorted(word_dict.items(), key=lambda x: x[1], reverse=True)
    # get the top 20 words
    top_words = sorted_dict[:20]
    # get the top 20 words and their frequency
    top_words, top_freq = zip(*top_words)
    # plot the top 20 words and their frequency
    plt.figure(figsize=(10, 5))
    plt.bar(top_words, top_freq)
    plt.title('Top 20 words in the dataset')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.show()

top_words(cleaned_df)
```



Inference: The word “the” has maximum occurrence.

Cleaned data

- The cleaned data set has been saved into a new file called cleaneddata3.csv

```
In [43]: # save list to csv file
         def save_list(without_stopwords, cleaneddataset1):
             # convert list to dataframe
             df = pd.DataFrame(without_stopwords)
             # save to csv file
             df.to_csv(cleaneddataset1, index= False)

In [44]: save_list(w, 'cleaneddataset3.csv')
```

Original Dataset - CSV FILE Cleaned Dataset - CSV FILE

Files

Notebooks-
https://drive.google.com/drive/folders/1vmwJ8V7kbAy1ikLui28vRP2LINtEI9ep?usp=sharing

Original Data Sets -
https://drive.google.com/drive/folders/1nS8pSDSveU6RmojQgEndhWxTmMzUpX0q?usp=sharing

Cleaned Data Sets -
https://drive.google.com/drive/folders/1kCGfVec42pasnecn8ixLRcmhMFFvqGEX?usp=sharing