```
# Import required libraries

import pandas as pd # require to work on dataframe and dataseries / specially used to work on data analysis and data manipulation
import numpy as np  # is fundamental python library used to perform mathematical operations
import nltk  # NLTK (Natural Language Toolkit) is a leading Python library for natural language processing (NLP) tasks.
import re # re is a built-in Python module that provides regular expression matching operations. Regular expressions are powerful patter
import string # provides various string-related functions and constants.
from nltk.corpus import stopwords
from nltk.corpus import stopwords # The stopwords corpus in NLTK is used to remove common words (stop words) from text data. Stop words
from nltk.tokenize import word_tokenize # Breaking text into individual words or tokens.
from sklearn.feature_extraction.text import CountVectorizer # The CountVectorizer class from the sklearn.feature_extraction.text module
from sklearn.model_selection import train_test_split # The train_test_split function from the sklearn.model_selection module in Python i
from sklearn.naive_bayes import MultinomialNB # The MultinomialNB class from the sklearn.naive_bayes module in Python is a probabilistic
```

```
""" NLTK (Natural Language Toolkit) is a leading Python library for natural language processing (NLP) tasks. It provides a collection of

Tokenization: Breaking text into individual words or tokens.
Stemming and lemmatization: Reducing words to their root form.
Part-of-speech tagging: Identifying the grammatical category of each word in a sentence (e.g., noun, verb, adjective).
Named entity recognition: Identifying named entities in text, such as people, organizations, and locations.
Parsing: Analyzing the grammatical structure of sentences.
Semantic analysis: Understanding the meaning of text.

Key features and applications of NLTK:

Text processing: NLTK provides tools for cleaning, preprocessing, and analyzing text data.
Language modeling: Building models to predict the next word or sequence of words in a text.
Machine translation: Translating text from one language to another.
Text summarization: Creating concise summaries of longer texts.
Sentiment analysis: Determining the sentiment expressed in a text (e.g., positive, negative, neutral).
Question answering: Answering questions based on a given text. """
```

```
' NLTK (Natural Language Toolkit) is a leading Python library for natural language processing (NLP) tasks. It provides a collection
of tools and resources for tasks such as:\n\nTokenization: Breaking text into individual words or tokens.\nStemming and lemmatizati
on: Reducing words to their root form.\nPart-of-speech tagging: Identifying the grammatical category of each word in a sentence (e.
g., noun, verb, adjective).\nNamed entity recognition: Identifying named entities in text, such as people, organizations, and locat
ions.\nParsing: Analyzing the grammatical structure of sentences.\nSemantic analysis: Understanding the meaning of text.\n\nKey fea
tures and applications of NLTK:\n\nText processing: NLTK provides tools for cleaning, preprocessing, and analyzing text data.\nLang
uage modeling: Building models to predict the next word or sequence of words in a text.\nMachine translation: Translating text from
one language to another.\nText summarization: Creating concise summaries of longer...'
```

```
df = pd.read_csv('/content/IMDB_Dataset.csv')
```

```
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape # Check the total records of the dataset
```

```
(50000, 2)
```

```
df.describe() # Check the statistical analysis of the dataset
```

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

```
df.info() # Check the information of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 781.4+ KB
```

```
df.isnull().sum() # Check the null values of the dataset
```

	0
review	0
sentiment	0

dtype: int64

```
df.sentiment.value_counts() # Check the unique values of the dataset
```

	count
sentiment	
positive	25000
negative	25000

dtype: int64

```
df.sentiment.replace({'positive':1,'negative':0}, inplace=True) # Replace the unique values of the dataset
```

```
<ipython-input-124-bb7cb35d91fe>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting 

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  df.sentiment.replace({'positive':1,'negative':0}, inplace=True) # Replace the unique values of the dataset
<ipython-input-124-bb7cb35d91fe>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future v
  df.sentiment.replace({'positive':1,'negative':0}, inplace=True) # Replace the unique values of the dataset
```

```
df.head()
```

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

Next steps: Generate code with df   View recommended plots   New interactive sheet

```
print(string.punctuation) # Print the string.punctuation
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```
import nltk
nltk.download('stopwords') # Download the stopwords
print(stopwords.words('english')) # Print the stopwords of the english language
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourse
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
df['review'] = df['review'].apply(lambda x:x.lower()) # Convert the text to lower case
```

```
df.head()
```

```
                                                review  sentiment
0   one of the other reviewers has mentioned that ...          1
1   a wonderful little production. <br /><br />the...          1
2   i thought this was a wonderful way to spend ti...          1
3       basically there's a family where a little boy ...      0
4   petter mattei's "love in the time of money" is...          1
```

Next steps: Generate code with df | View recommended plots | New interactive sheet

Suggested code may be subject to a licence | AbderrhmanAbdellatif/Fake-News-Detection | M-Tallal-Habib/Label-Generation-for-Textual-Data-using-Unsupervised-Learning | ShivankUdayawal/NEWS-

```
# Removal of HTML strips and noise text

from bs4 import BeautifulSoup

def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

# Removing the sqaure brackets

def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

# Removing the noisy text

def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    return text

# Apply function on review column

df['review']=df['review'].apply(denoise_text)
```

```
<ipython-input-130-e3801900136d>:6: MarkupResemblesLocatorWarning: The input looks more like a filename than markup. You may want tc
  soup = BeautifulSoup(text, "html.parser")
```

Suggested code may be subject to a licence | techillasingh/pydatascience

```
# Define the function to remove special characters

def remove_special_characters(text, remove_digits=True):
    pattern=r'[^a-zA-Z0-9\s]'
    text=re.sub(pattern,'',text)
    return text

# Apply the function on review column

df['review']=df['review'].apply(remove_special_characters)


# Remove the repeatative words

df['review'] = df['review'].apply(lambda x: ' '.join([word for word in x.split() if len(word)>2]))


# Remvoing the stopwords from the dataset

stop = stopwords.words('english')
df['review'] = df['review'].apply(lambda x:' '.join([word for word in x.split() if word not in (stop)]))


df.head()
```

```
                                               review  sentiment
0      one reviewers mentioned watching episode youll...      1
1          wonderful little production filming technique ...  1
2  thought wonderful way spend time hot summer we...          1
3            basically theres family little boy jake thinks...  0
4        petter matteis love time money visually stunni...    1
```

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
 # Separate out words to apply tokenization in the dataset
df['review'] = df['review'].apply(lambda x: x.split())
```

```
df.head()
```

	review	sentiment
0	[one, reviewers, mentioned, watching, episode,...	1
1	[wonderful, little, production, filming, techn...	1
2	[thought, wonderful, way, spend, time, hot, su...	1
3	[basically, theres, family, little, boy, jake,...	0
4	[petter, matteis, love, time, money, visually,...	1

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Apply stemming on the dataset

from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

df['review'] = df['review'].apply(lambda x: [stemmer.stem(i) for i in x])
```

```
df.head()
```

	review	sentiment
0	[one, review, mention, watch, episod, youll, h...	1
1	[wonder, littl, product, film, techniqu, unass...	1
2	[thought, wonder, way, spend, time, hot, summe...	1
3	[basic, there, famili, littl, boy, jake, think...	0
4	[petter, mattei, love, time, money, visual, st...	1

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Now lets swtich these stemming together

for i in range(len(df['review'])):
  df['review'][i]=' '.join(df['review'][i])
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
  df['review'][i]=' '.join(df['review'][i])
<ipython-input-139-b5989f2da827>:4: FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases, but when using Copy-on-Write (which wil
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single step and ensure this keeps updating the

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers

  df['review'][i]=' '.join(df['review'][i])
<ipython-input-139-b5989f2da827>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
  df['review'][i]=' '.join(df['review'][i])
```

```
df.head()
```

	review	sentiment
0	one review mention watch episod youll hook rig...	1
1	wonder littl product film techniqu unassum old...	1
2	thought wonder way spend time hot summer weeke...	1
3	basic there famili littl boy jake think there ...	0
4	petter mattei love time money visual stun film...	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Apply the TfidfVectorizer on the dataset

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
tkidf = TfidfVectorizer(max_features=20000, ngram_range=(1,3),analyzer='char')
```

Double-click (or enter) to edit

```
X = tkidf.fit_transform(df['review'])
y = df['sentiment']
```

```
X.shape
```

```
(50000, 19946)
```

```
y.shape
```

```
(50000,)
```

```
# Perform Data Sampling

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

```
# Apply the Linear SVC model
clf = LinearSVC()
clf.fit(X_train, y_train)
```

```
▾ LinearSVC
LinearSVC()
```

```
y_pred = clf.predict(X_test) # Predict on test dataset
```

```
# Print the classification report
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.86      0.86      4979
           1       0.86      0.88      0.87      5021

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000
```

```
# Accuracy of the model

accuracy_score(y_test, y_pred)*100
```

```
86.56
```

```
# Apply the Second model -  MultinomialNB

NB = MultinomialNB()
NB.fit(X_train, y_train)
```

```
▾  MultinomialNB ⓘ ?
MultinomialNB()
```

```
y_nb_pred = NB.predict(X_test) # Predict on test dataset
```

```
# Print the Classification report based on MultinomialNB model

print(classification_report(y_test, y_nb_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.81      0.81      4979
           1       0.81      0.83      0.82      5021

    accuracy                           0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.82      0.82      0.82     10000
```

```
# Print the accuracy of the model

accuracy_score(y_test, y_nb_pred)*100
```

```
81.6
```

```
# Apply the random forest classifier

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
▾  RandomForestClassifier ⓘ ?
RandomForestClassifier()
```

```
y_pred_rf = rf.predict(X_test) # Predict on test dataset
```

```
# Print the classification report

print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.78      0.78      0.78      4979
           1       0.78      0.78      0.78      5021

    accuracy                           0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.78      0.78      0.78     10000
```

```
# Accuracy of the model
```

```
accuracy_score(y_test, y_pred_rf)*100
```

```
78.24
```

```
# From the above model, Linear SVC model has better accuracy than Random Forest and MultinomialNB model. Hence we will be going forward
```

```
# Confusion matrix of Linear SVC model

confusion_matrix(y_test, y_pred)
```

```
array([[4258,  721],
       [ 623, 4398]])
```

```
Start coding or generate with AI.
```

```
# Apply and check Linear SVC model on some example

x='I loved movie'
vec  =tkidf.transform([x])
clf.predict(vec)
```

```
array([1])
```

```
# As we can see model comes under array=1, thence this sentence has positive sentiment
```

```
 x  = 'product is bad, high quality'
 vec = tkidf.transform([x])
 clf.predict(vec)
```

```
array([0])
```

```
 x  = 'I love this movie, but now bored'
 vec = tkidf.transform([x])
 clf.predict(vec)
```

```
array([0])
```

```
 x  = 'dont wanna it'
 vec = tkidf.transform([x])
 clf.predict(vec)
```

```
array([0])
```

```
"""Understanding the common words used in the tweets
 Now I want to see how well the given sentiments are distributed across the train dataset. One
 way to accomplish this task is by understanding the common words by plotting wordclouds.
 A wordcloud is a visualization wherein the most frequent words appear in large size and the less
 frequent words appear in smaller sizes.
 Let’s visualize all the words our data using the wordcloud plot."""
```

```
'Understanding the common words used in the tweets\n Now I want to see how well the given sentiments are distributed across the tra
in dataset. One\n way to accomplish this task is by understanding the common words by plotting wordclouds.\n A wordcloud is a visua
lization wherein the most frequent words appear in large size and the less\n frequent words appear in smaller sizes.\n Let’s visual
ize all the words our data using the wordcloud plot.'
```

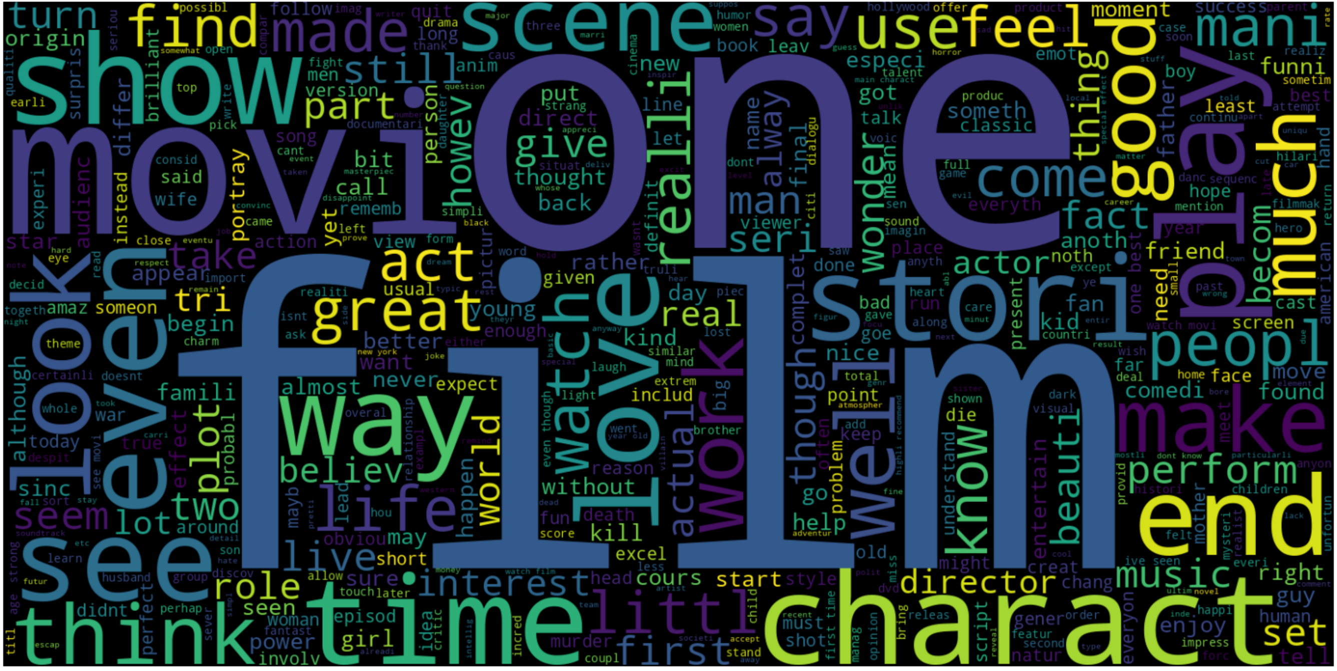
```
# Import the required libraries

from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Positive Words

pos_words = ' '.join([text for text in df['review'][df['sentiment']==1]])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(pos_words)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most frequent words in positive review', fontsize=19)
plt.show()
```

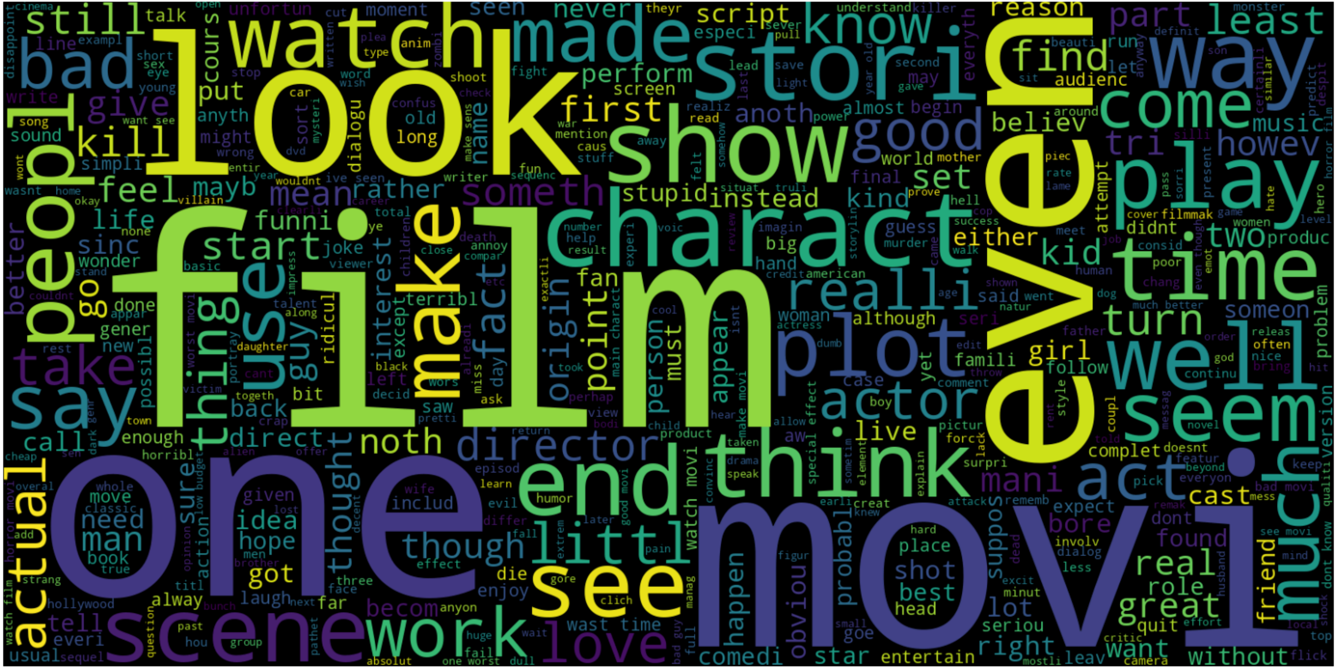
Most frequent words in positive review



```
# Ngeative Words

pos_words = ' '.join([text for text in df['review'][df['sentiment']==0]])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(pos_words)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most frequent words in negative review', fontsize=19)
plt.show()
```

Most frequent words in negative review



Start coding or generate with AI.

```
# Let's calculate the Positivity and Subjectivity from the dataset

""" The first one is Polarity, which indicates the positivity/negativity in the sentiment of the text. The second one is subjectivity wh
```

```
' The first one is Polarity, which indicates the positivity/negativity in the sentiment of the text. The second one is subjectivity
which refers to objective info/facts versus personal opinions or emotions.'
```

Suggested code may be subject to a licence | wahid028/Sentiment-Analysis | AradhyaMahant/Crypto-Price-Prediction-webapp |

```
from textblob import TextBlob

pol = lambda x: TextBlob(x).sentiment.polarity
sub = lambda x: TextBlob(x).sentiment.subjectivity

df['polarity'] = df['review'].apply(pol)
df['subjectivity'] = df['review'].apply(sub)
```

```
df.head()
```

	review	sentiment	polarity	subjectivity
0	one review mention watch episod youll hook rig...	1	0.006566	0.454900
1	wonder littl product film techniqu unassum old...	1	0.235000	0.235000
2	thought wonder way spend time hot summer weeke...	1	0.347143	0.527143
3	basic there famili littl boy jake think there ...	0	-0.008333	0.484722
4	petter mattei love time money visual stun film...	1	0.193900	0.321292

Next steps: Generate code with df View recommended plots New interactive sheet

Start coding or generate with AI.

```
""" Thank you """
```

```
' Thank you '
```

Start coding or generate with AI.

Start coding or generate with AI.