In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
style.use('ggplot')
import re
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split


```

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

In [3]:
```
df.shape
```

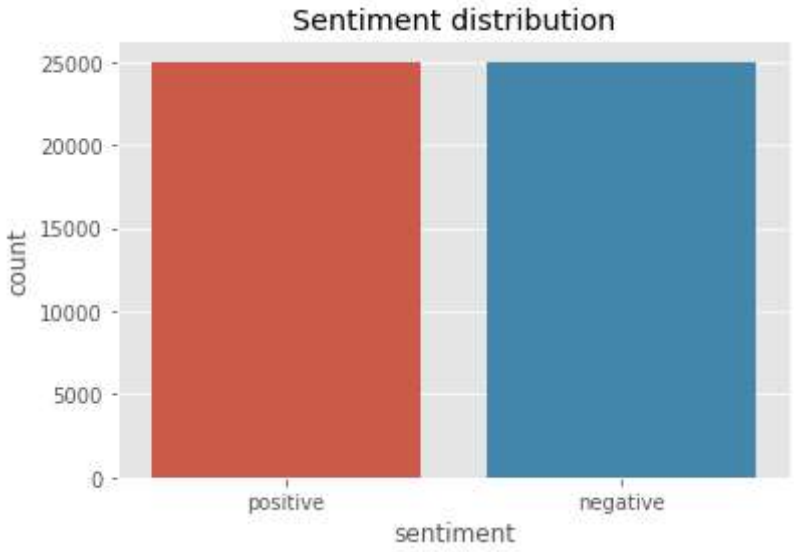
Out[3]:
```
(50000, 2)
```

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

```
<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
```

In [5]:
```
1 sns.countplot(x='sentiment', data=df)
2 plt.title("Sentiment distribution")
```

Out[5]: Text(0.5, 1.0, 'Sentiment distribution')



In [6]:
```
for i in range(5):
    print("Review: ", [i])
    print(df['review'].iloc[i], "\n")
    print("Sentiment: ", df['sentiment'].iloc[i], "\n\n")
```

```
Review:  [0]
One of the other reviewers has mentioned that after watching just 1 Oz episode
you'll be hooked. They are right, as this is exactly what happened with me.<br
/><br />The first thing that struck me about Oz was its brutality and unflinchi
ng scenes of violence, which set in right from the word GO. Trust me, this is n
ot a show for the faint hearted or timid. This show pulls no punches with regar
ds to drugs, sex or violence. Its is hardcore, in the classic use of the word.<
br /><br />It is called OZ as that is the nickname given to the Oswald Maximum
Security State Penitentary. It focuses mainly on Emerald City, an experimental
section of the prison where all the cells have glass fronts and face inwards, s
o privacy is not high on the agenda. Em City is home to many..Aryans, Muslims,
gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death s
tares, dodgy dealings and shady agreements are never far away.<br /><br />I wou
ld say the main appeal of the show is due to the fact that it goes where other
shows wouldn't dare. Forget pretty pictures painted for mainstream audiences, f
orget charm, forget romance...OZ doesn't mess around. The first episode I ever
saw struck me as so nasty it was surreal, I couldn't say I was ready for it, bu
t as I watched more, I developed a taste for Oz, and got accustomed to the high
levels of graphic violence. Not just violence, but injustice (crooked guards wh
o'll be sold out for a nickel, inmates who'll kill on order and get away with i
t, well mannered, middle class inmates being turned into prison bitches due to
their lack of street skills or prison experience) Watching Oz, you may become c
omfortable with what is uncomfortable viewing....thats if you can get in touch
with your darker side.

Sentiment:  positive


Review:  [1]
A wonderful little production. <br /><br />The filming technique is very unassu
ming- very old-time-BBC fashion and gives a comforting, and sometimes discomfor
ting, sense of realism to the entire piece. <br /><br />The actors are extremel
y well chosen- Michael Sheen not only "has got all the polari" but he has all t
he voices down pat too! You can truly see the seamless editing guided by the re
ferences 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 on
e of the great master's of comedy and his life. <br /><br />The realism really
comes home with the little things: the fantasy of the guard which, rather than
use the traditional 'dream' techniques remains solid then disappears. It plays
on our knowledge and our senses, particularly with the scenes concerning Orton
and Halliwell and the sets (particularly of their flat with Halliwell's murals
decorating every surface) are terribly well done.

Sentiment:  positive


Review:  [2]
I thought this was a wonderful way to spend time on a too hot summer weekend, s
itting in the air conditioned theater and watching a light-hearted comedy. The
plot is simplistic, but the dialogue is witty and the characters are likable (e
ven the well bread suspected serial killer). While some may be disappointed whe
n they realize this is not Match Point 2: Risk Addiction, I thought it was proo
```

```
f that Woody Allen is still fully 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's comedies in
years (dare I say a decade?). While I've never been impressed with Scarlet Joha
nson, in this she managed to tone down her "sexy" image and jumped right into a
average, but spirited young woman.<br /><br />This may not be the crown jewel o
f his career, but it was wittier than "Devil Wears Prada" and more interesting
than "Superman" a great comedy to go see with friends.

Sentiment:  positive


Review:  [3]
Basically there's a family where a little boy (Jake) thinks there's a zombie in
his closet & his parents are fighting all the time.<br /><br />This movie is sl
ower than a soap opera... and suddenly, Jake decides to become Rambo and kill t
he zombie.<br /><br />OK, first of all when you're going to make a film you mus
t Decide if its a thriller or a drama! As a drama the movie is watchable. Paren
ts are divorcing & arguing like in real life. And then we have Jake with his cl
oset which totally ruins all the film! I expected to see a BOOGEYMAN similar mo
vie, and instead i watched a drama with some meaningless thriller spots.<br /><
br />3 out of 10 just for the well playing parents & descent dialogs. As for th
e shots with Jake: just ignore them.

Sentiment:  negative


Review:  [4]
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watc
h. Mr. Mattei offers us a vivid portrait about human relations. This is a movie
that seems to be telling us what money, power and success do to people in the d
ifferent situations we encounter. <br /><br />This being a variation on the Art
hur Schnitzler's play about the same theme, the director transfers the action t
o the present time New York where all these different characters meet and conne
ct. Each one is connected in one way, or another to the next person, but no one
seems to know the previous point of contact. Stylishly, the film has a sophisti
cated luxurious look. We are taken to see how these people live and the world t
hey live in their own habitat.<br /><br />The only thing one gets out of all th
ese souls in the picture is the different stages of loneliness each one inhabit
s. A big city is not exactly the best place in which human relations find since
re fulfillment, as one discerns is the case with most of the people we encounte
r.<br /><br />The acting is good under Mr. Mattei's direction. Steve Buscemi, R
osario Dawson, Carol Kane, Michael Imperioli, Adrian Grenier, and the rest of t
he talented cast, make these characters come alive.<br /><br />We wish Mr. Matt
ei good luck and await anxiously for his next work.

Sentiment:  positive
```

In [7]:

```
1 def no_of_words(text):
2     words= text.split()
3     word_count = len(words)
4     return word_count
```

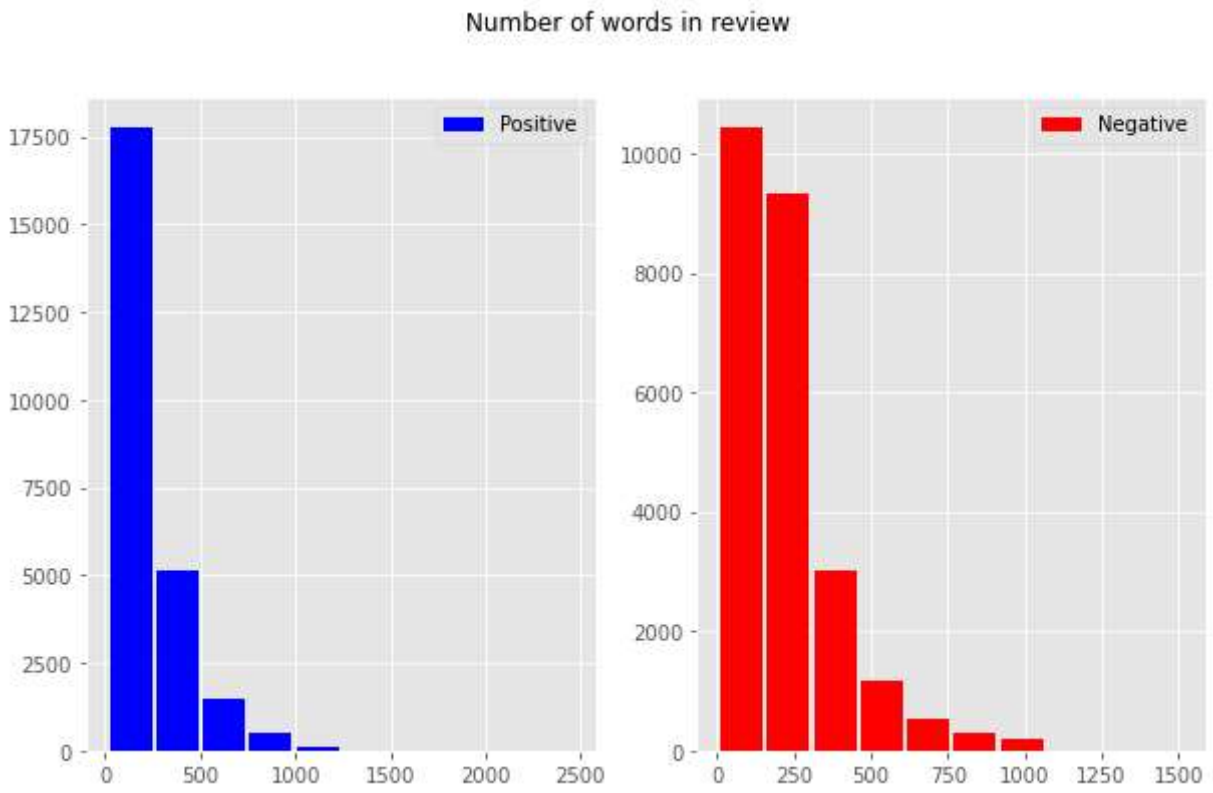
In [8]:
```
df['word count'] = df['review'].apply(no_of_words)
```

In [9]:
```
df.head()
```

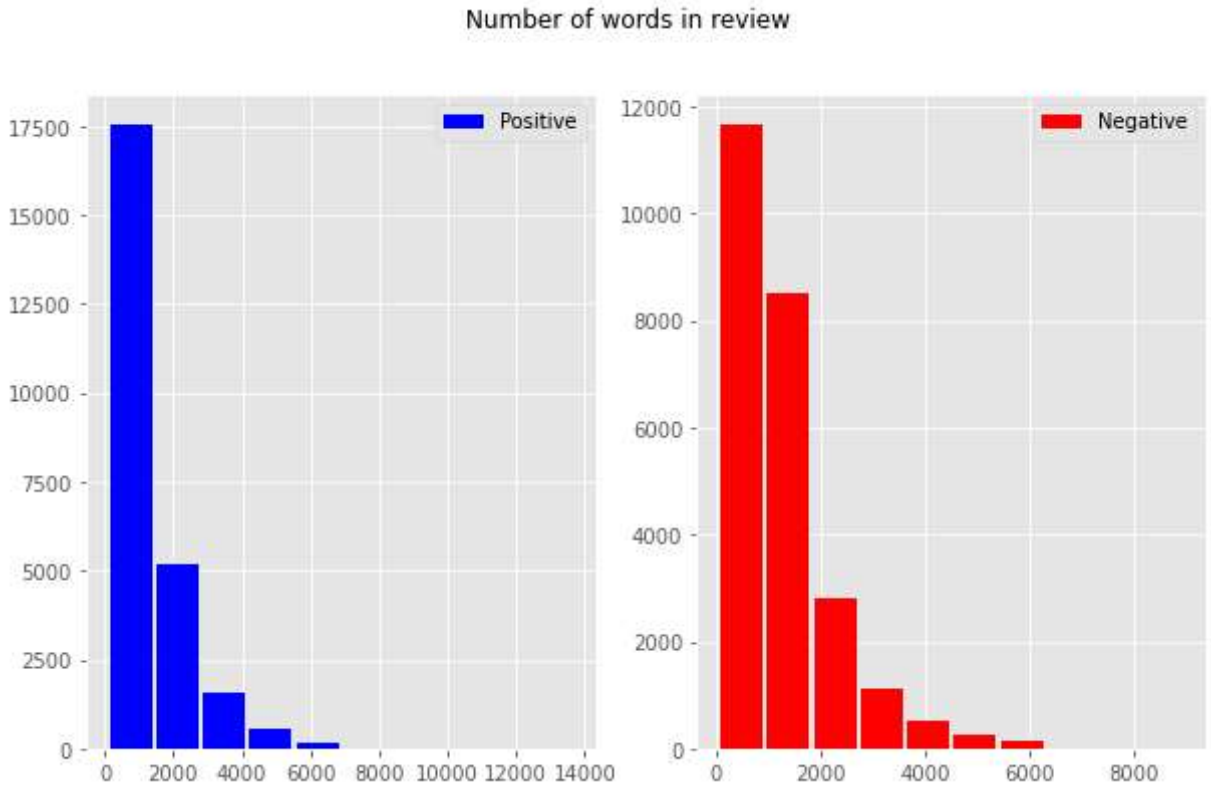
Out[9]:

	review	sentiment	word count
0	One of the other reviewers has mentioned that ...	positive	307
1	A wonderful little production. The...	positive	162
2	I thought this was a wonderful way to spend ti...	positive	166
3	Basically there's a family where a little boy ...	negative	138
4	Petter Mattei's "Love in the Time of Money" is...	positive	230

In [10]:
```
fig, ax = plt.subplots(1,2, figsize=(10,6))
ax[0].hist(df[df['sentiment'] == 'positive']['word count'], label='Positive'
ax[0].legend(loc='upper right');
ax[1].hist(df[df['sentiment'] == 'negative']['word count'], label='Negative'
ax[1].legend(loc='upper right');
fig.suptitle("Number of words in review")
plt.show()
```



In [11]:
```
fig, ax = plt.subplots(1,2, figsize=(10,6))
ax[0].hist(df[df['sentiment'] == 'positive']['review'].str.len(), label='Pos
ax[0].legend(loc='upper right');
ax[1].hist(df[df['sentiment'] == 'negative']['review'].str.len(), label='Neg
ax[1].legend(loc='upper right');
fig.suptitle("Number of words in review")
plt.show()
```



In [12]:
```
df.sentiment.replace("positive", 1, inplace=True)
df.sentiment.replace("negative", 2, inplace=True)
```

In [13]:
```
df.head()
```

Out[13]:

	review	sentiment	word count
0	One of the other reviewers has mentioned that ...	1	307
1	A wonderful little production. The...	1	162
2	I thought this was a wonderful way to spend ti...	1	166
3	Basically there's a family where a little boy ...	2	138
4	Petter Mattei's "Love in the Time of Money" is...	1	230

In [14]:
```
def data_processing(text):
    text= text.lower()
    text = re.sub('<br />', '', text)
    text = re.sub(r"https\S+|www\S+|http\S+", '', text, flags = re.MULTILINE
    text = re.sub(r'\@w+|\#', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text_tokens = word_tokenize(text)
    filtered_text = [w for w in text_tokens if not w in stop_words]
    return " ".join(filtered_text)
```

In [16]:
```
duplicated_count = df.duplicated().sum()
print("Number of duplicate entries: ", duplicated_count)
```

```
Number of duplicate entries:  418
```

In [17]:
```
df = df.drop_duplicates('review')
```

In [18]:
```
stemmer = PorterStemmer()
def stemming(data):
    text = [stemmer.stem(word) for word in data]
    return data
```

In [19]:
```
df.review = df['review'].apply(lambda x: stemming(x))
```

In [20]:
```
df['word count'] = df['review'].apply(no_of_words)
df.head()
```

Out[20]:

	review	sentiment	word count
0	One of the other reviewers has mentioned that ...	1	307
1	A wonderful little production. The...	1	162
2	I thought this was a wonderful way to spend ti...	1	166
3	Basically there's a family where a little boy ...	2	138
4	Petter Mattei's "Love in the Time of Money" is...	1	230

In [21]:
```
pos_reviews =  df[df.sentiment == 1]
pos_reviews.head()
```

Out[21]:

	review	sentiment	word count
0	One of the other reviewers has mentioned that ...	1	307
1	A wonderful little production. The...	1	162
2	I thought this was a wonderful way to spend ti...	1	166
4	Petter Mattei's "Love in the Time of Money" is...	1	230
5	Probably my all-time favorite movie, a story o...	1	119

In [23]:
```
from collections import Counter
count = Counter()
for text in pos_reviews['review'].values:
    for word in text.split():
        count[word] +=1
count.most_common(15)
```

Out[23]:
```
[('the', 290932),
 ('and', 165372),
 ('a', 155251),
 ('of', 148673),
 ('to', 127921),
 ('is', 107829),
 ('in', 90145),
 ('that', 62191),
 ('I', 61914),
 ('it', 53162),
 ('this', 51403),
 ('/><br', 48800),
 ('as', 46419),
 ('with', 43153),
 ('was', 41934)]
```

In [24]:
```
pos_words = pd.DataFrame(count.most_common(15))
pos_words.columns = ['word', 'count']
pos_words.head()
```

Out[24]:

	word	count
0	the	290932
1	and	165372
2	a	155251
3	of	148673
4	to	127921

In [26]:
```
neg_reviews =  df[df.sentiment == 2]
neg_reviews.head()
```

Out[26]:

	review	sentiment	word count
3	Basically there's a family where a little boy ...	2	138
7	This show was an amazing, fresh & innovative i...	2	174
8	Encouraged by the positive comments about this...	2	130
10	Phil the Alien is one of those quirky films wh...	2	96
11	I saw this movie when I was about 12 when it c...	2	180

In [28]:
```
count = Counter()
for text in neg_reviews['review'].values:
    for word in text.split():
        count[word] +=1
count.most_common(15)
```

Out[28]:
```
[('the', 273542),
 ('a', 149568),
 ('and', 134388),
 ('of', 132924),
 ('to', 131974),
 ('is', 93846),
 ('in', 78593),
 ('I', 69449),
 ('that', 63687),
 ('this', 61414),
 ('it', 53956),
 ('/><br', 51411),
 ('was', 49969),
 ('for', 39373),
 ('with', 38797)]
```

In [29]:
```
neg_words = pd.DataFrame(count.most_common(15))
neg_words.columns = ['word', 'count']
neg_words.head()
```

Out[29]:

	word	count
0	the	273542
1	a	149568
2	and	134388
3	of	132924
4	to	131974

In [31]:
```
X = df['review']
Y = df['sentiment']
```

In [32]:
```
vect = TfidfVectorizer()
X = vect.fit_transform(df['review'])
```

In [33]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, ran
```

In [34]:
```
print("Size of x_train: ", (x_train.shape))
print("Size of y_train: ", (y_train.shape))
print("Size of x_test: ", (x_test.shape))
print("Size of y_test: ", (y_test.shape))
```

```
Size of x_train:  (34707, 101895)
Size of y_train:  (34707,)
Size of x_test:  (14875, 101895)
Size of y_test:  (14875,)
```

In [44]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion
import warnings
warnings.filterwarnings('ignore')
```

In [47]:
```
mnb = MultinomialNB()
mnb.fit(x_train, y_train)
mnb_pred = mnb.predict(x_test)
mnb_acc = accuracy_score(mnb_pred, y_test)
print("Test accuracy: {:.2f}%".format(mnb_acc*100))
```

```
Test accuracy: 86.11%
```

In [48]:
```
print(confusion_matrix(y_test, mnb_pred))
print("\n")
print(classification_report(y_test, mnb_pred))
```

```
[[6276 1195]
 [ 871 6533]]


              precision    recall  f1-score   support

           1       0.88      0.84      0.86      7471
           2       0.85      0.88      0.86      7404

    accuracy                           0.86     14875
   macro avg       0.86      0.86      0.86     14875
weighted avg       0.86      0.86      0.86     14875
```

In [49]:
```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 89.60%
```

In [50]:
```
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

```
[[6771  700]
 [ 847 6557]]


              precision    recall  f1-score   support

           1       0.89      0.91      0.90      7471
           2       0.90      0.89      0.89      7404

    accuracy                           0.90     14875
   macro avg       0.90      0.90      0.90     14875
weighted avg       0.90      0.90      0.90     14875
```