In [5]:
```
# pip install WordCloud
```

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
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 wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In [1], line 12
     10 from nltk.corpus import stopwords
     11 stop_words=set(stopwords.words('english'))
---> 12 from wordcloud import WordCloud
     13 from sklearn.feature_extraction.text import TfidfVectorizer
     14 from sklearn.model_selection import train_test_split

ModuleNotFoundError: No module named 'wordcloud'
```

In [ ]:
```
# pip install plotly
```

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

In [8]:
```
df.head()
```

Out[8]:

	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

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

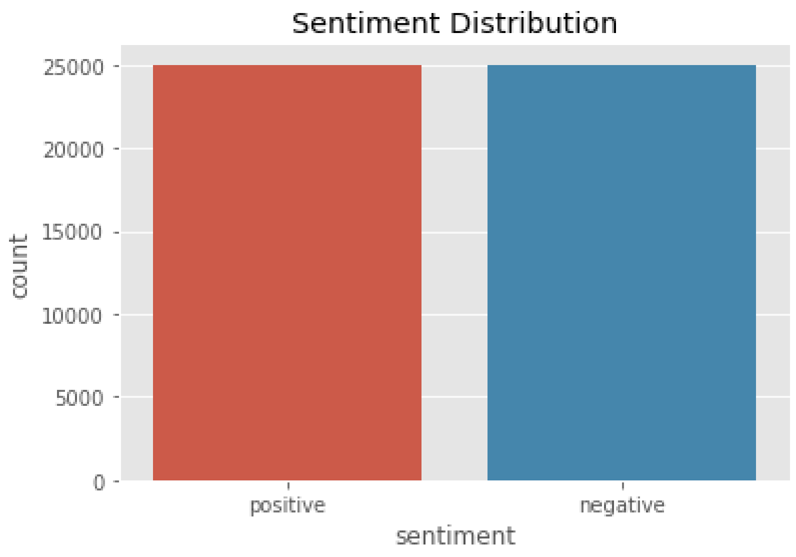
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

In [10]:
```
df.shape
```

Out[10]:
```
(50000, 2)
```

```
In [11]: sns.countplot(x='sentiment',data=df)
         plt.title('Sentiment Distribution')
```

```
Out[11]: Text(0.5, 1.0, 'Sentiment Distribution')
```



Reading of first five review

```
In [13]: 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 yo
u'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 unflinching scenes
of violence, which set in right from the word GO. Trust me, this is not a show for
the faint hearted or timid. This show pulls no punches with regards to drugs, sex
or violence. Its is hardcore, in the classic use of the word.<br /><br />It is cal
led OZ as that is the nickname given to the Oswald Maximum Security State Penitent
ary. It focuses mainly on Emerald City, an experimental section of the prison wher
e all the cells have glass fronts and face inwards, so privacy is not high on the
agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, I
talians, Irish and more....so scuffles, death stares, dodgy dealings and shady agr
eements are never far away.<br /><br />I would say the main appeal of the show is
due to the fact that it goes where other shows wouldn't dare. Forget pretty pictur
es painted for mainstream audiences, forget charm, forget romance...OZ doesn't mes
s around. The first episode I ever saw struck me as so nasty it was surreal, I cou
ldn't say I was ready for it, but as I watched more, I developed a taste for Oz, a
nd got accustomed to the high levels of graphic violence. Not just violence, but i
njustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on o
rder and get away with it, well mannered, middle class inmates being turned into p
rison bitches due to their lack of street skills or prison experience) Watching O
z, you may become comfortable with what is uncomfortable viewing....thats if you c
an get in touch with your darker side.

Sentiment:  positive


Review [1]
A wonderful little production. <br /><br />The filming technique is very unassumin
g- very old-time-BBC fashion and gives a comforting, and sometimes discomforting,
sense of realism to the entire piece. <br /><br />The actors are extremely well ch
osen- Michael Sheen not only "has got all the polari" but he has all the voices do
wn pat too! You can truly see the seamless editing guided by the references to Wil
liams' diary entries, not only is it well worth the watching but it is a terrificl
y written and performed piece. A masterful production about one of the great maste
r's of comedy and his life. <br /><br />The realism really comes home with the lit
tle things: the fantasy of the guard which, rather than use the traditional 'drea
m' techniques remains solid then disappears. It plays on our knowledge and our sen
ses, particularly with the scenes concerning Orton and Halliwell and the sets (par
ticularly of their flat with Halliwell's murals decorating every surface) are terr
ibly well done.

Sentiment:  positive


Review [2]
I thought this was a wonderful way to spend time on a too hot summer weekend, sitt
ing in the air conditioned theater and watching a light-hearted comedy. The plot i
s simplistic, but the dialogue is witty and the characters are likable (even the w
ell bread suspected serial killer). While some may be disappointed when they reali
ze this is not Match Point 2: Risk Addiction, I thought it was proof that Woody Al
len 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 Johanson, in this she manag
ed to tone down her "sexy" image and jumped right into a average, but spirited you
ng woman.<br /><br />This may not be the crown jewel of his career, but it was wit
tier 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 hi
```

```
s 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 zombi
e.<br /><br />OK, first of all when you're going to make a film you must Decide if
its a thriller or a drama! As a drama the movie is watchable. Parents are divorcin
g & arguing like in real life. And then we have Jake with his closet which totally
ruins all the film! I expected to see a BOOGEYMAN similar movie, and instead i wat
ched a drama with some meaningless thriller spots.<br /><br />3 out of 10 just for
the well playing parents & descent dialogs. As for the shots with Jake: just ignor
e them.

Sentiment:  negative


Review [4]
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch.
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 different
situations we encounter. <br /><br />This being a variation on the Arthur Schnitzl
er's play about the same theme, the director transfers the action to the present t
ime New York where all these different characters meet and connect. Each one is co
nnected in one way, or another to the next person, but no one seems to know the pr
evious point of contact. Stylishly, the film has a sophisticated luxurious look. W
e are taken to see how these people live and the world they live in their own habi
tat.<br /><br />The only thing one gets out of all these souls in the picture is t
he different stages of loneliness each one inhabits. A big city is not exactly the
best place in which human relations find sincere fulfillment, as one discerns is t
he case with most of the people we encounter.<br /><br />The acting is good under
Mr. Mattei's direction. Steve Buscemi, Rosario Dawson, Carol Kane, Michael Imperio
li, Adrian Grenier, and the rest of the talented cast, make these characters come
alive.<br /><br />We wish Mr. Mattei good luck and await anxiously for his next wo
rk.

Sentiment:  positive
```

In [13]:
```python
# Find number of word in each review
```

In [14]:
```python
def no_of_words(text):
    words=text.split()
    word_count=len(words)
    return word_count
```

In [15]:
```python
df['Word Count']=df['review'].apply(no_of_words)
```

In [16]:
```python
df.head()
```

Out[16]:

	review	sentiment	Word Count
0	One of the other reviewers has mentioned that ...	positive	307
1	A wonderful little production. <br /><br />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

Now convert target columns into numerical formats

```
In [18]: df.sentiment.replace('positive',1,inplace=True)
         df.sentiment.replace('negative',0,inplace=True)
```

```
In [20]: df
```

Out[20]:

	review	sentiment	Word Count
0	One of the other reviewers has mentioned that ...	1	307
1	A wonderful little production. <br /><br />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 ...	0	138
4	Petter Mattei's "Love in the Time of Money" is...	1	230
...	...	...	...
49995	I thought this movie did a down right good job...	1	194
49996	Bad plot, bad dialogue, bad acting, idiotic di...	0	112
49997	I am a Catholic taught in parochial elementary...	0	230
49998	I'm going to have to disagree with the previou...	0	212
49999	No one expects the Star Trek movies to be high...	0	129

50000 rows × 3 columns

```
In [21]: # Text processing for conversion.Apply data processing functions to remove punctat
```

```
In [22]: def data_preprocessing(text):
             text=text.lower()
             text=re.sub('<br />','',text)
             text=re.sub(r'http\S+','',text,flags=re.MULTILINE)
             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 [23]: import nltk
         nltk.download('punkt')

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\admin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Out[23]: True
```

```
In [24]: df.review=df['review'].apply(data_preprocessing)
```

```
In [25]: df['Word Count']=df['review'].apply(no_of_words)
```

```
In [26]: df
```

Out[26]:

	review	sentiment	Word Count
0	one reviewers mentioned watching 1 oz episode ...	1	168
1	wonderful little production filming technique ...	1	84
2	thought wonderful way spend time hot summer we...	1	86
3	basically theres family little boy jake thinks...	0	67
4	petter matteis love time money visually stunni...	1	125
...	...	...	...
49995	thought movie right good job wasnt creative or...	1	85
49996	bad plot bad dialogue bad acting idiotic direc...	0	55
49997	catholic taught parochial elementary schools n...	0	115
49998	im going disagree previous comment side maltin...	0	115
49999	one expects star trek movies high art fans exp...	0	68

50000 rows × 3 columns

In [27]:
```
duplicated_count=df.duplicated().sum()
print("Number of duplicated entries:",duplicated_count)
```

```
Number of duplicated entries: 422
```

In [28]:
```
df=df.drop_duplicates('review')
```

In [29]:
```
df
```

Out[29]:

	review	sentiment	Word Count
0	one reviewers mentioned watching 1 oz episode ...	1	168
1	wonderful little production filming technique ...	1	84
2	thought wonderful way spend time hot summer we...	1	86
3	basically theres family little boy jake thinks...	0	67
4	petter matteis love time money visually stunni...	1	125
...	...	...	...
49995	thought movie right good job wasnt creative or...	1	85
49996	bad plot bad dialogue bad acting idiotic direc...	0	55
49997	catholic taught parochial elementary schools n...	0	115
49998	im going disagree previous comment side maltin...	0	115
49999	one expects star trek movies high art fans exp...	0	68

49578 rows × 3 columns

In [30]:
```
duplicated_count=df.duplicated().sum()
print("Number of duplicated entries:",duplicated_count)
```

```
Number of duplicated entries: 0
```

Perform Stemming on text data

In [31]:
```
stemmer=PorterStemmer()
def stemming (data):
    text=[stemmer.stem(word) for word in data]
    return data
```

In [32]:
```
df.review=df['review'].apply(lambda x:stemming(x))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:5516: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

Now chek number of word after preprocessing

In [33]:
```
df['Word Count']=df['review'].apply(no_of_words)
df.head()
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_14932/1144763576.py:1: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Word Count']=df['review'].apply(no_of_words)
```

Out[33]:

	review	sentiment	Word Count
0	one reviewers mentioned watching 1 oz episode ...	1	168
1	wonderful little production filming technique ...	1	84
2	thought wonderful way spend time hot summer we...	1	86
3	basically theres family little boy jake thinks...	0	67
4	petter matteis love time money visually stunni...	1	125

Split data for model bulding

In [19]:
```
X=df['review']
Y=df['sentiment']
```

In [20]:
```
vect=TfidfVectorizer()
X=vect.fit_transform(df['review'])
```

In [26]:
```
X
```

```
Out[26]: <50000x101895 sparse matrix of type '<class 'numpy.float64'>'
         	with 6826529 stored elements in Compressed Sparse Row format>
```

```
In [27]: X[0]
```

```
Out[27]: <1x101895 sparse matrix of type '<class 'numpy.float64'>'
         	with 186 stored elements in Compressed Sparse Row format>
```

```
In [28]: Y
```

```
Out[28]: 0        1
         1        1
         2        1
         3        0
         4        1
                 ..
         49995    1
         49996    0
         49997    0
         49998    0
         49999    0
         Name: sentiment, Length: 50000, dtype: int64
```

```
In [ ]:
```

```
In [29]: x_train,x_test,y_train,y_test=train_test_split(X,Y,train_size=0.7,random_state=42)
```

```
In [30]: 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 :  (35000, 101895)
Size of y_train :  (35000,)
Size of x_test :  (15000, 101895)
Size of y_test :  (15000,)
```

Reduce data size for demonstartion as it will face problem of processing on hardware

```
In [31]: x_train=x_train[:10000, :10000]
         y_train=y_train[:10000]
         x_test=x_test[:2000, :10000]
         y_test=y_test[:2000]
```

```
In [32]: 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 :  (10000, 10000)
Size of y_train :  (10000,)
Size of x_test :  (2000, 10000)
Size of y_test :  (2000,)
```

In [33]:
```
x_train_new=x_train
y_train_new=y_train
x_test_new=x_test
y_test_new=y_test
```

In [34]:
```
print("Size of x_train_new : ",(x_train_new.shape))
print("Size of y_train_new : ",(y_train_new.shape))
print("Size of x_test_new: ",(x_test_new.shape))
print("Size of y_test_new : ",(y_test_new.shape))
```

```
Size of x_train_new :  (10000, 10000)
Size of y_train_new :  (10000,)
Size of x_test_new:  (2000, 10000)
Size of y_test_new :  (2000,)
```

In [35]:
```
x_train_new
```

Out[35]:
```
<10000x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 157200 stored elements in Compressed Sparse Row format>
```

In [36]:
```
y_train_new
```

Out[36]:
```
38094    0
40624    1
49425    0
35734    1
41708    0
        ..
7110     1
46643    1
5440     1
33017    0
47138    0
Name: sentiment, Length: 10000, dtype: int64
```

In [37]:
```
x_train_new=x_train_new.toarray()
x_test_new=x_test_new.toarray()
```

In [38]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [39]:
```
model=Sequential()
model.add(Dense(units=16,activation='relu',input_dim=x_train_new.shape[1]))
model.add(Dense(units=8,activation='relu'))
model.add(Dense(units=1,activation='sigmoid'))
```

In [40]:
```
model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

In [41]:
```
x_train_new
```

Out[41]:
```
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.]])
```

In [42]:
```
y_train_new
```

Out[42]:
```
38094    0
40624    1
49425    0
35734    1
41708    0
        ..
7110     1
46643    1
5440     1
33017    0
47138    0
Name: sentiment, Length: 10000, dtype: int64
```

In [43]:
```
history=model.fit(x_train_new,y_train_new,batch_size=10,epochs=15)
```

```
Epoch 1/15
1000/1000 [==============================] - 3s 2ms/step - loss: 0.6392 - accurac
y: 0.6488
Epoch 2/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.5274 - accurac
y: 0.7450
Epoch 3/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4981 - accurac
y: 0.7613
Epoch 4/15
1000/1000 [==============================] - 1s 1ms/step - loss: 0.4849 - accurac
y: 0.7692
Epoch 5/15
1000/1000 [==============================] - 1s 1ms/step - loss: 0.4769 - accurac
y: 0.7756
Epoch 6/15
1000/1000 [==============================] - 2s 1ms/step - loss: 0.4703 - accurac
y: 0.7796
Epoch 7/15
1000/1000 [==============================] - 1s 1ms/step - loss: 0.4646 - accurac
y: 0.7812
Epoch 8/15
1000/1000 [==============================] - 1s 1ms/step - loss: 0.4596 - accurac
y: 0.7856
Epoch 9/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4562 - accurac
y: 0.7876
Epoch 10/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4517 - accurac
y: 0.7879
Epoch 11/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4489 - accurac
y: 0.7896
Epoch 12/15
1000/1000 [==============================] - 2s 1ms/step - loss: 0.4457 - accurac
y: 0.7924
Epoch 13/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4432 - accurac
y: 0.7928
Epoch 14/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4412 - accurac
y: 0.7981
Epoch 15/15
1000/1000 [==============================] - 2s 2ms/step - loss: 0.4394 - accurac
y: 0.7956
```

In [44]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                160016

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 160,161
Trainable params: 160,161
Non-trainable params: 0
_________________________________________________________________
```

In [45]:
```
test_loss, test_acc = model.evaluate(x_test_new, y_test_new)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.5220 - accuracy: 0.
7430
```

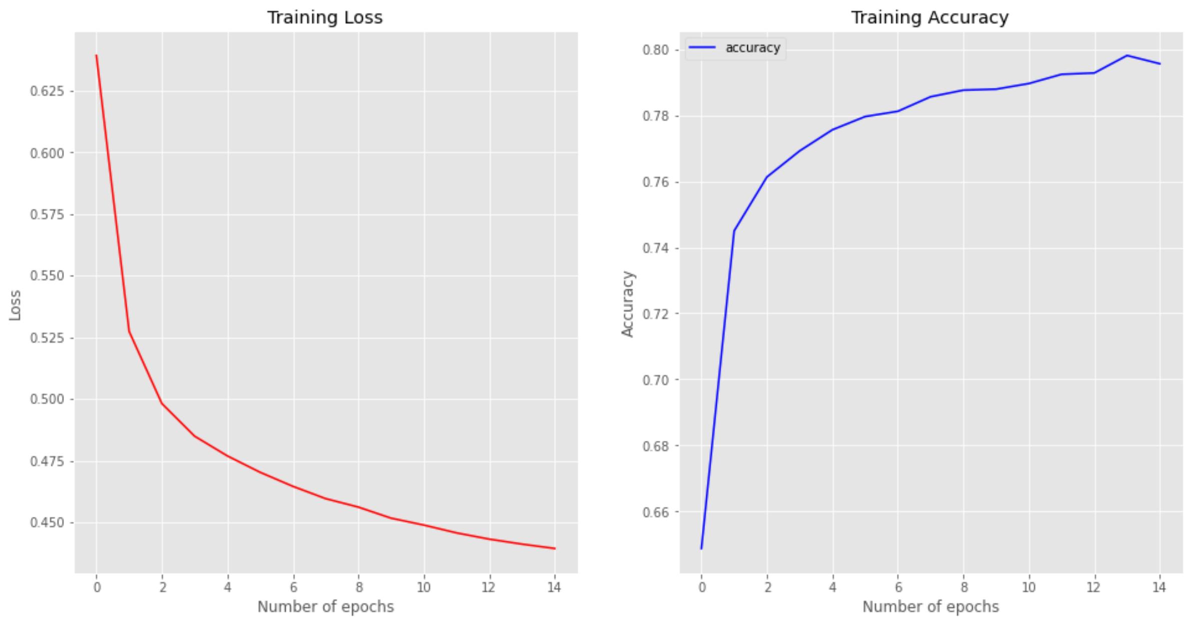
In [46]:
```
print("test_loss: ", test_loss)
```

```
test_loss:  0.5220049023628235
```

In [47]:
```
print("test_acc: ", test_acc)
```

```
test_acc:  0.7429999709129333
```

In [48]:
```
plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plt.plot(history.history['loss'], color='r', label='loss')
plt.title('Training Loss')
plt.xlabel("Number of epochs")
plt.ylabel("Loss")
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], color='b', label='accuracy')
plt.title('Training Accuracy')
plt.xlabel("Number of epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



```
In [ ]: Similarly we can check the performance on different optimizer 'adam'
```