NAME : SOHAM R SHAH
SEAT N0: 46
SUBJECT: ML_MODEL ASSINGMENT

```
import nltk
nltk.download('stopwords')

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
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
df = pd.read_csv('IMDB Dataset.csv')
df.head()
```

	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

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

```
df.shape
```

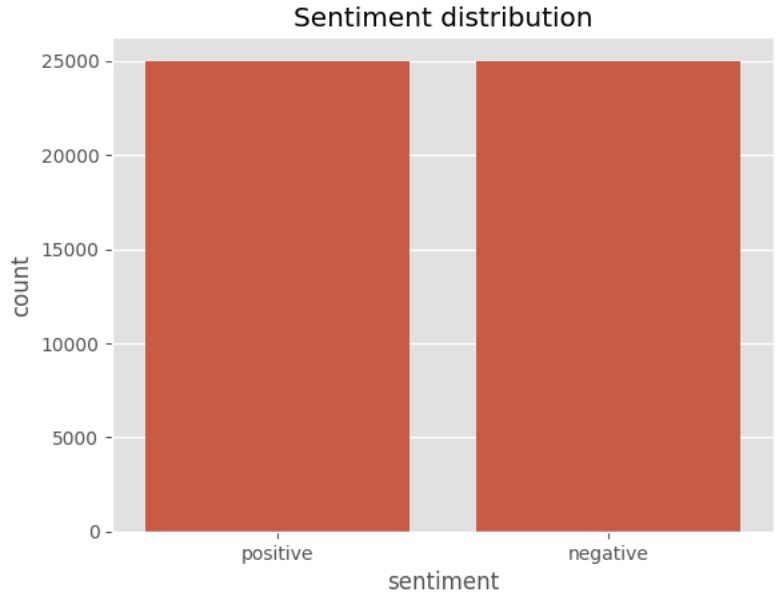
```
(50000, 2)
```

```
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
```

```
sns.countplot(x='sentiment', data=df)
plt.title("Sentiment distribution")
```

```
Text(0.5, 1.0, 'Sentiment distribution')
```




```
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

Sentiment:  positive


Review:  [1]
A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashion and gives a c

Sentiment:  positive


Review:  [2]
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and wat

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 th

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 abou

Sentiment:  positive
```

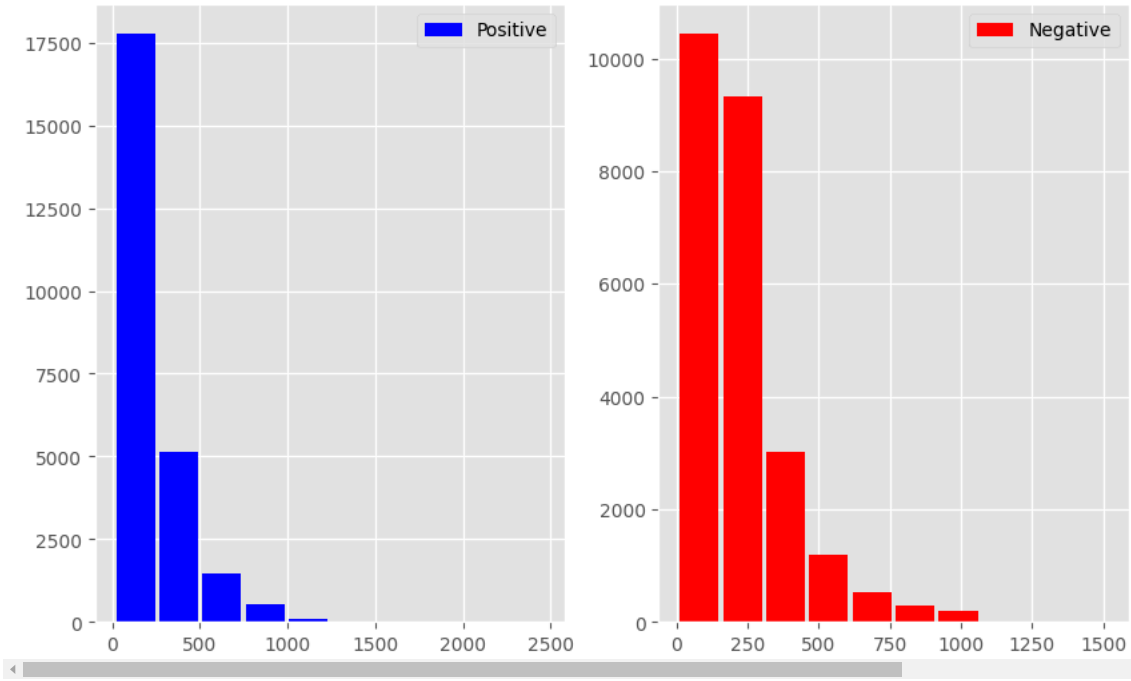
```
def no_of_words(text):
    words= text.split()
    word_count = len(words)
    return word_count


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

```
df.head()
```

	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

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

```
fig, ax = plt.subplots(1,2, figsize=(10,6))
ax[0].hist(df[df['sentiment'] == 'positive']['word count'], label='Positive', color='blue', rwidth=0.9);
ax[0].legend(loc='upper right');
ax[1].hist(df[df['sentiment'] == 'negative']['word count'], label='Negative', color='red', rwidth=0.9);
ax[1].legend(loc='upper right');
fig.suptitle("Number of words in review")
plt.show()
```




```
df.sentiment.replace("positive", 1, inplace=True)
df.sentiment.replace("negative", 2, inplace=True)
```

```
<ipython-input-13-7cef2d76bcd1>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

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


  df.sentiment.replace("positive", 1, inplace=True)
<ipython-input-13-7cef2d76bcd1>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are

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


  df.sentiment.replace("negative", 2, inplace=True)
<ipython-input-13-7cef2d76bcd1>:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a f
  df.sentiment.replace("negative", 2, inplace=True)
```

```
df.head()
```

	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

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

```
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)
```

```
!pip install nltk
import nltk
nltk.download('punkt_tab') # Download the required resource for tokenization
```

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

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

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

```
Number of duplicate entries:  421
```

```
df = df.drop_duplicates('review')
```

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

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

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

	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...	2	67
4	petter matteis love time money visually stunni...	1	125

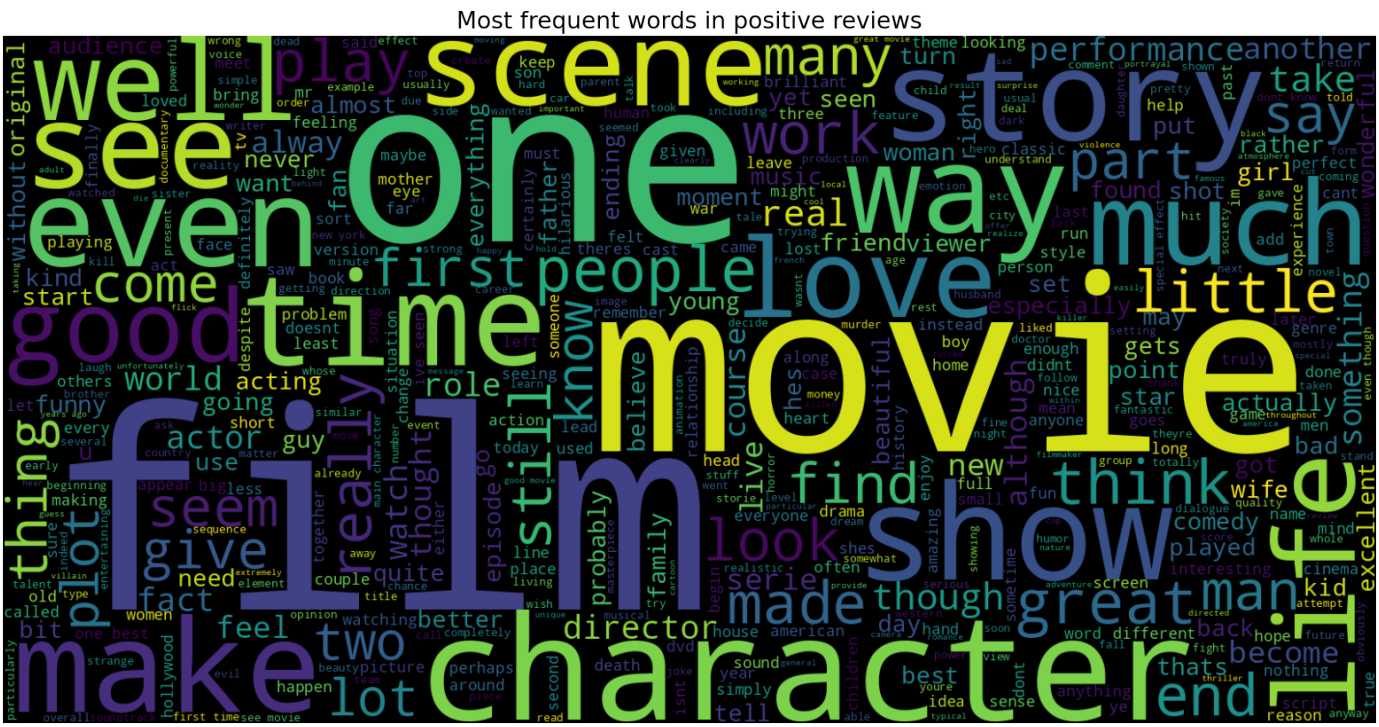
Next steps: Generate code with df | View recommended plots | New interactive sheet

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

	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
4	petter matteis love time money visually stunni...	1	125
5	probably alltime favorite movie story selfless...	1	58

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

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




```
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)
```

```
[('film', 39285),
 ('movie', 35830),
 ('one', 25621),
 ('like', 16998),
 ('good', 14281),
 ('great', 12568),
 ('story', 12338),
 ('see', 11814),
 ('time', 11724),
 ('well', 10930),
 ('really', 10638),
 ('also', 10516),
```

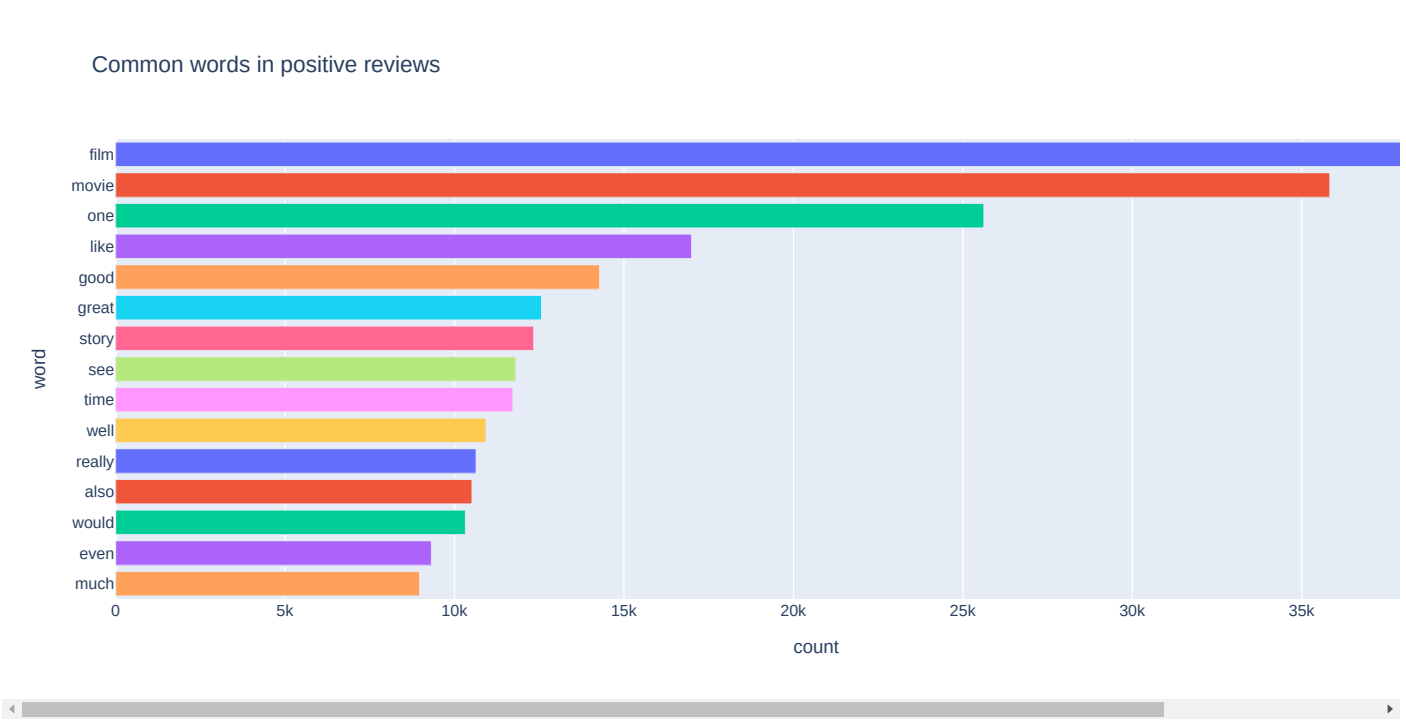
```
('would', 10320),
('even', 9318),
('much', 8971)]
```

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

	word	count
0	film	39285
1	movie	35830
2	one	25621
3	like	16998
4	good	14281

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

```
px.bar(pos_words, x='count', y='word', title='Common words in positive reviews', color = 'word')
```




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

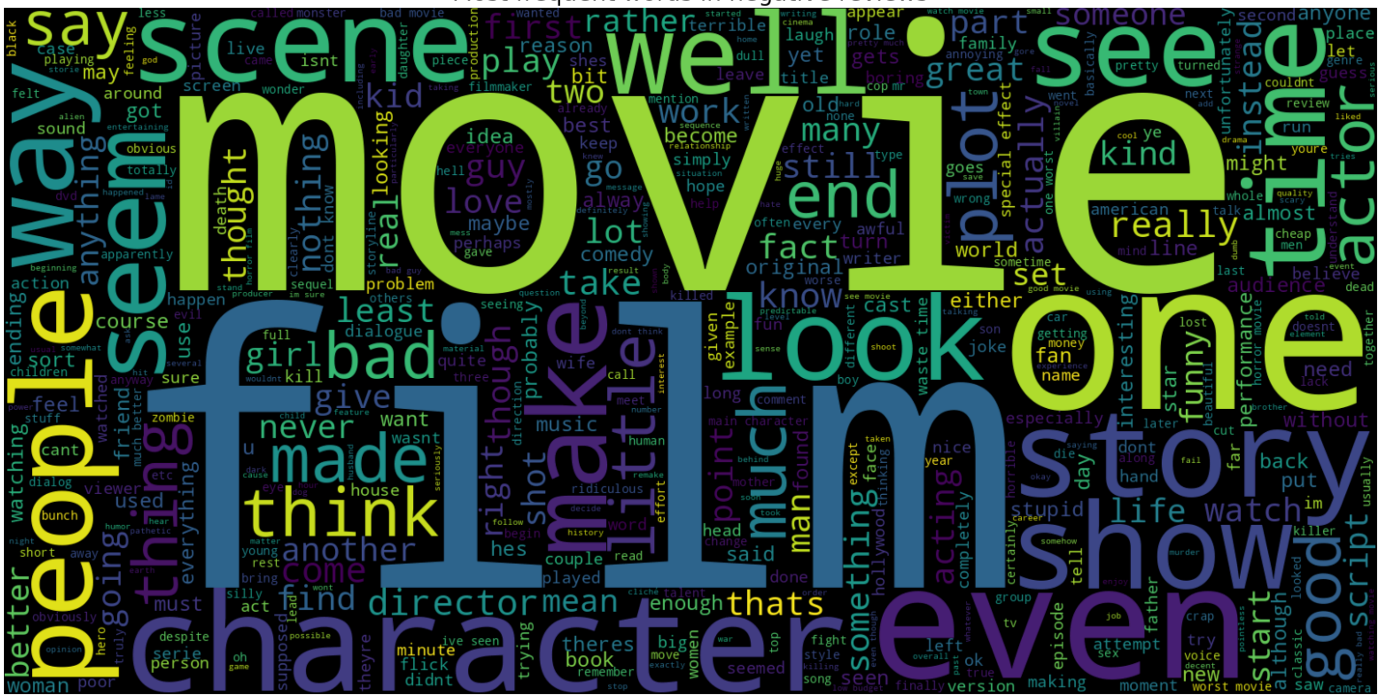
	review	sentiment	word count
3	basically theres family little boy jake thinks...	2	67
7	show amazing fresh innovative idea 70s first a...	2	83
8	encouraged positive comments film looking forw...	2	64
10	phil alien one quirky films humour based aroun...	2	51
11	saw movie 12 came recall scariest scene big bi...	2	84

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

```
text = ' '.join([word for word in neg_reviews['review']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

```
plt.title('Most frequent words in negative reviews', fontsize = 19)
plt.show()
```

Most frequent words in negative reviews




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

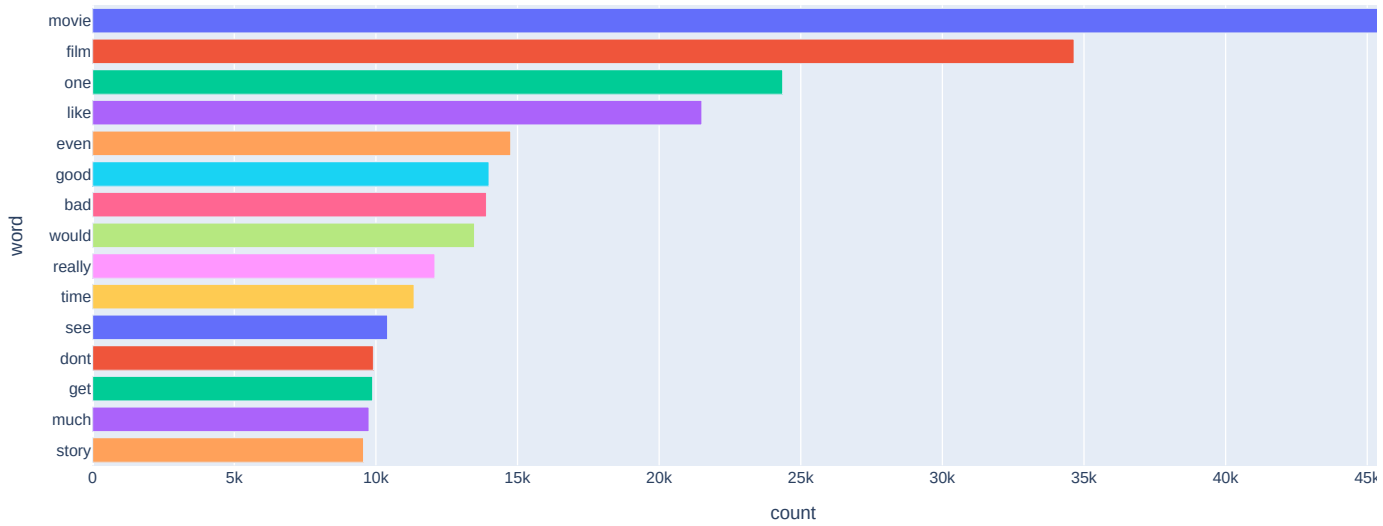
```
[('movie', 47001),
 ('film', 34651),
 ('one', 24361),
 ('like', 21508),
 ('even', 14759),
 ('good', 13995),
 ('bad', 13903),
 ('would', 13482),
 ('really', 12084),
 ('time', 11349),
 ('see', 10412),
 ('dont', 9912),
 ('get', 9884),
 ('much', 9758),
 ('story', 9563)]
```

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

	word	count
0	movie	47001
1	film	34651
2	one	24361
3	like	21508
4	even	14759

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

```
px.bar(neg_words, x='count', y='word', title='Common words in negative reviews', color = 'word')
```




```
X = df['review']
Y = df['sentiment']
```

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

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

```
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:  (34704, 221707)
Size of y_train:  (34704,)
Size of x_test:  (14874, 221707)
Size of y_test:  (14874,)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

```
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: 89.06%
```

```
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

```
[[6790  723]
 [ 904 6457]]
```

```
                  precision    recall  f1-score   support

               1       0.88      0.90      0.89      7513
               2       0.90      0.88      0.89      7361

        accuracy                           0.89     14874
       macro avg       0.89      0.89      0.89     14874
    weighted avg       0.89      0.89      0.89     14874
```

```
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: 86.44%
```

```
print(confusion_matrix(y_test, mnb_pred))
print("\n")
print(classification_report(y_test, mnb_pred))
```

```
[[6439 1074]
 [ 943 6418]]


                  precision    recall  f1-score   support

               1       0.87      0.86      0.86      7513
               2       0.86      0.87      0.86      7361

        accuracy                           0.86     14874
       macro avg       0.86      0.86      0.86     14874
    weighted avg       0.86      0.86      0.86     14874
```

```
svc = LinearSVC()
svc.fit(x_train, y_train)
svc_pred = svc.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("Test accuracy: {:.2f}%".format(svc_acc*100))
```

```
Test accuracy: 89.22%
```

```
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
```

```
[[6766  747]
 [ 857 6504]]


                  precision    recall  f1-score   support

               1       0.89      0.90      0.89      7513
               2       0.90      0.88      0.89      7361

        accuracy                           0.89     14874
       macro avg       0.89      0.89      0.89     14874
    weighted avg       0.89      0.89      0.89     14874
```

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C':[0.1, 1, 10, 100], 'loss':['hinge', 'squared_hinge']}
grid = GridSearchCV(svc, param_grid, refit=True, verbose = 3)
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV 1/5] END .................C=0.1, loss=hinge;, score=0.872 total time=   3.1s
[CV 2/5] END .................C=0.1, loss=hinge;, score=0.875 total time=   2.0s
[CV 3/5] END .................C=0.1, loss=hinge;, score=0.871 total time=   0.7s
[CV 4/5] END .................C=0.1, loss=hinge;, score=0.878 total time=   0.7s
[CV 5/5] END .................C=0.1, loss=hinge;, score=0.874 total time=   0.4s
[CV 1/5] END .........C=0.1, loss=squared_hinge;, score=0.892 total time=   0.9s
[CV 2/5] END .........C=0.1, loss=squared_hinge;, score=0.895 total time=   0.7s
[CV 3/5] END .........C=0.1, loss=squared_hinge;, score=0.888 total time=   0.7s
[CV 4/5] END .........C=0.1, loss=squared_hinge;, score=0.894 total time=   0.7s
[CV 5/5] END .........C=0.1, loss=squared_hinge;, score=0.890 total time=   0.7s
[CV 1/5] END ...................C=1, loss=hinge;, score=0.896 total time=   1.5s
[CV 2/5] END ...................C=1, loss=hinge;, score=0.894 total time=   3.5s
[CV 3/5] END ...................C=1, loss=hinge;, score=0.893 total time=   2.1s
[CV 4/5] END ...................C=1, loss=hinge;, score=0.894 total time=   1.1s
[CV 5/5] END ...................C=1, loss=hinge;, score=0.894 total time=   1.2s
[CV 1/5] END ...........C=1, loss=squared_hinge;, score=0.892 total time=   0.8s
[CV 2/5] END ...........C=1, loss=squared_hinge;, score=0.895 total time=   0.7s
[CV 3/5] END ...........C=1, loss=squared_hinge;, score=0.889 total time=   0.7s
[CV 4/5] END ...........C=1, loss=squared_hinge;, score=0.896 total time=   0.8s
[CV 5/5] END ...........C=1, loss=squared_hinge;, score=0.894 total time=   0.8s
[CV 1/5] END ..................C=10, loss=hinge;, score=0.875 total time=   6.6s
[CV 2/5] END ..................C=10, loss=hinge;, score=0.882 total time=  15.6s
[CV 3/5] END ..................C=10, loss=hinge;, score=0.875 total time=   8.3s
[CV 4/5] END ..................C=10, loss=hinge;, score=0.881 total time=   8.2s
[CV 5/5] END ..................C=10, loss=hinge;, score=0.878 total time=   8.2s
[CV 1/5] END ..........C=10, loss=squared_hinge;, score=0.881 total time=   2.7s
[CV 2/5] END ..........C=10, loss=squared_hinge;, score=0.885 total time=   2.5s
[CV 3/5] END ..........C=10, loss=squared_hinge;, score=0.879 total time=   4.1s
[CV 4/5] END ..........C=10, loss=squared_hinge;, score=0.885 total time=   3.3s
[CV 5/5] END ..........C=10, loss=squared_hinge;, score=0.883 total time=   3.3s
[CV 1/5] END .................C=100, loss=hinge;, score=0.876 total time=   4.5s
[CV 2/5] END .................C=100, loss=hinge;, score=0.881 total time=  15.1s
[CV 3/5] END .................C=100, loss=hinge;, score=0.874 total time=  16.6s
[CV 4/5] END .................C=100, loss=hinge;, score=0.880 total time=   7.2s
[CV 5/5] END .................C=100, loss=hinge;, score=0.878 total time=  15.5s
[CV 1/5] END .........C=100, loss=squared_hinge;, score=0.877 total time=   3.1s
[CV 2/5] END .........C=100, loss=squared_hinge;, score=0.882 total time=   9.6s
[CV 3/5] END .........C=100, loss=squared_hinge;, score=0.875 total time=  10.3s
[CV 4/5] END .........C=100, loss=squared_hinge;, score=0.881 total time=  10.1s
[CV 5/5] END .........C=100, loss=squared_hinge;, score=0.878 total time=  10.8s
```

▸ **GridSearchCV**

▸ **best_estimator_: LinearSVC**

▾ LinearSVC

```
LinearSVC(C=1, loss='hinge')
```

```
print("best cross validation score: {:.2f}".format(grid.best_score_))
print("best parameters: ", grid.best_params_)
```

```
best cross validation score: 0.89
best parameters:  {'C': 1, 'loss': 'hinge'}
```

```
svc = LinearSVC(C = 1, loss='hinge')
svc.fit(x_train, y_train)
svc_pred = svc.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("Test accuracy: {:.2f}%".format(svc_acc*100))
```

```
Test accuracy: 89.41%
```

```
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
```

```
[[6788  725]
 [ 850 6511]]


              precision    recall  f1-score   support

           1       0.89      0.90      0.90      7513
           2       0.90      0.88      0.89      7361

    accuracy                           0.89     14874
```

```
           macro avg       0.89      0.89      0.89     14874
        weighted avg       0.89      0.89      0.89     14874
```