```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import re
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

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

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

```
data = pd.read_csv('/content/flipkart_data.csv')
data.head()
```

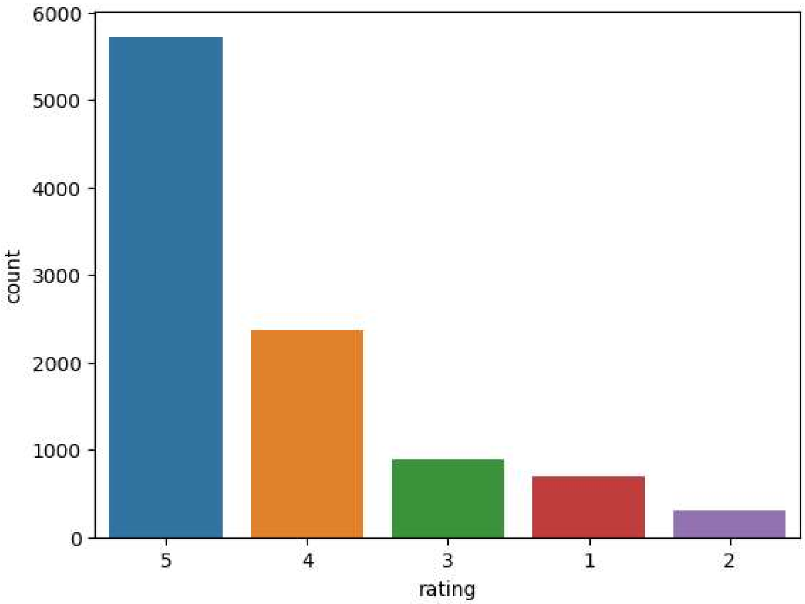
| | review | rating |
|---|---|---|
| **0** | It was nice produt. I like it's design a lot. ... | 5 |
| **1** | awesome sound....very pretty to see this nd th... | 5 |
| **2** | awesome sound quality. pros 7-8 hrs of battery... | 4 |
| **3** | I think it is such a good product not only as ... | 5 |
| **4** | awesome bass sound quality very good bettary l... | 5 |

```
# unique ratings
pd.unique(data['rating'])
```

```
array([5, 4, 1, 3, 2])
```

```
sns.countplot(data=data,
        x='rating',
        order=data.rating.value_counts().index)
```

```
<Axes: xlabel='rating', ylabel='count'>
```



```
# rating label(final)
pos_neg = []
for i in range(len(data['rating'])):
    if data['rating'][i] >= 5:
        pos_neg.append(1)
    else:
        pos_neg.append(0)

data['label'] = pos_neg
```

```
from tqdm import tqdm


def preprocess_text(text_data):
    preprocessed_text = []

    for sentence in tqdm(text_data):
        # Removing punctuations
        sentence = re.sub(r'[^\w\s]', '', sentence)

        # Converting lowercase and removing stopwords
        preprocessed_text.append(' '.join(token.lower()
                                          for token in nltk.word_tokenize(sentence)
                                          if token.lower() not in stopwords.words('english')))

    return preprocessed_text
```

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

from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

with open('your_file.txt', 'r', encoding='utf-8') as file:
    text_data = file.read()

# Now text_data is a regular string, and you can apply string-based operations on it.


def preprocess_text(text_data):
    # Tokenize the text
    tokens = word_tokenize(text_data)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

    # Stem the tokens
    stemmer = SnowballStemmer('english')
    stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

    # Join the stemmed tokens into a preprocessed text
    preprocessed_text = ' '.join(stemmed_tokens)

    return preprocessed_text

# Example usage:
preprocessed_review = preprocess_text(data['review'].values)
data['review'] = preprocessed_review
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-2420aec6c280> in <cell line: 26>()
     24 
     25 # Example usage:
---> 26 preprocessed_review = preprocess_text(data['review'].values)
     27 data['review'] = preprocessed_review

                                  10 frames
/usr/local/lib/python3.10/dist-packages/nltk/tokenize/punkt.py in
_match_potential_end_contexts(self, text)
   1393         previous_slice = slice(0, 0)
   1394         previous_match = None
-> 1395         for match in self._lang_vars.period_context_re().finditer(text):
```

```
data.head()
```

| | review | rating | label |
|---|---|---|---|
| 0 | It was nice produt. I like it's design a lot. ... | 5 | 1 |
| 1 | awesome sound....very pretty to see this nd th... | 5 | 1 |
| 2 | awesome sound quality. pros 7-8 hrs of battery... | 4 | 0 |
| 3 | I think it is such a good product not only as ... | 5 | 1 |
| 4 | awesome bass sound quality very good bettary l... | 5 | 1 |

```
data["label"].value_counts()
```

```
1    5726
0    4250
Name: label, dtype: int64
```

```
consolidated = ' '.join(
    word for word in data['review'][data['label'] == 1].astype(str))
wordCloud = WordCloud(width=1600, height=800,
                      random_state=21, max_font_size=110)
plt.figure(figsize=(15, 10))
plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
cv = TfidfVectorizer(max_features=2500)
X = cv.fit_transform(data['review'] ).toarray()
```

```
X
```

```
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.]])
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, data['label'],
                                                    test_size=0.33,
                                                    stratify=data['label'],
                                                    random_state = 42)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

model = DecisionTreeClassifier(random_state=0)
model.fit(X_train,y_train)

#testing the model
pred = model.predict(X_train)
print(accuracy_score(y_train,pred))
```

```
0.9362561723776747
```

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=0)
model.fit(X_train,y_train)

#testing the model
pred = model.predict(X_train)
print(accuracy_score(y_train,pred))
```

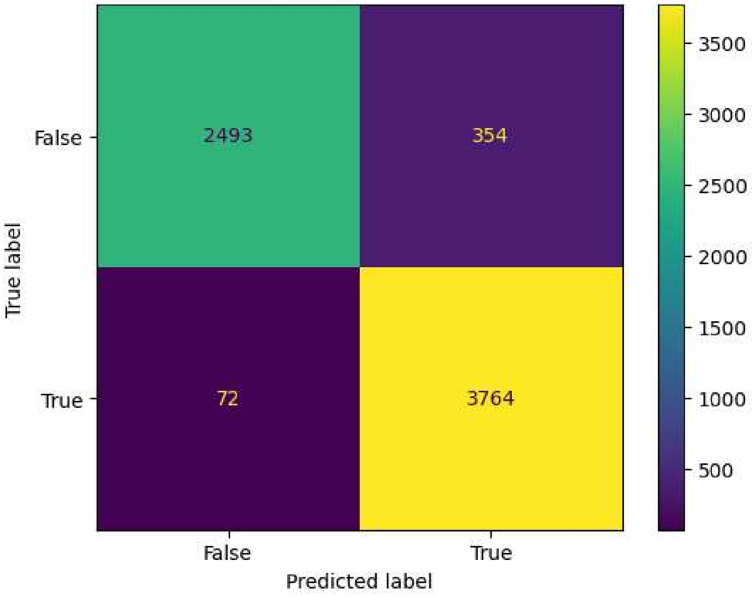
```
0.9362561723776747
```

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_train,pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm,
                      display_labels = [False, True])

cm_display.plot()
plt.show()
```


False
True
True label
2493
354
72
3764
False
True
Predicted label
3500
3000
2500
2000
1500
1000
500