```
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential

df =
pd.read_csv("/content/drive/MyDrive/online_sas/online_review.csv")

df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2304,\n  \"fields\":
[\n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 665,\n        \"min\": 0,\n
\"max\": 2303,\n        \"num_unique_values\": 2304,\n
\"samples\": [\n          1640,\n          508,\n          1422\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Product_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 231,\n        \"samples\": [\n          \"LG 24
inch Full HD LED Backlit IPS Panel Monitor (24MP400)\\u00a0\\
u00a0(Response Time: 5 ms)\",\n          \"LG 260 L Frost Free Double
Door Top Mount 3 Star Convertible Refrigerator\\u00a0\\u00a0(Dazzle
Steel, GL-S292RDSX)\",\n          \"HP Ryzen 3 Dual Core 3250U - (8
GB/256 GB SSD/Windows 10 Home) 15s-GY0501AU Thin and Light Laptop\\
u00a0\\u00a0(15.6 inch, Natural Silver, 1.69 kg, With MS Office)\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Review\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1358,\n        \"samples\": [\n          \"Im statisfied .. valueble
money\",\n          \"Nice product nice design but not big actually
same 7.5 kg size...\",\n          \"awesom ips led monitor\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rating\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          4,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2304 entries, 0 to 2303
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   Unnamed: 0    2304 non-null   int64
 1   Product_name  2304 non-null   object
 2   Review        2304 non-null   object
 3   Rating        2304 non-null   int64
dtypes: int64(2), object(2)
memory usage: 72.1+ KB
```

```
null_values = df.isnull().sum()
print("Null values in the entire Data:")
print(null_values)
```

```
Null values in the entire Data:
Unnamed: 0      0
Product_name    0
Review          0
Rating          0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
null_values = df.isnull().sum()
null_values
```

```
Unnamed: 0      0
Product_name    0
Review          0
Rating          0
dtype: int64
```

```
df.drop_duplicates(inplace=True)
```

```
import string
df['Review'] = df['Review'].apply(lambda x: x.lower())
df['Review'] = df['Review'].apply(lambda x:
x.translate(str.maketrans('', '',
string.punctuation)))
```

```
 df['Review']
```

```
0       best under 60k great performancei got it for a...
1                                         good perfomence
2       great performance but usually it has also that...
3                my wife is so happy and best product 😘
4       light weight laptop with new amazing features ...
                              ...
2299    great display accurate colours at this price r...
2300    superb monitor first brought 1 used for 2 mont...
2301                                              awesome
2302                          only one issue with adapter
2303    worth the money u spend for this monitor great...
Name: Review, Length: 2304, dtype: object
```

```
from sklearn.feature_extraction.text import CountVectorizer
# Assuming 'df' is your Data containing text data
text_data = df['Review']
vectorizer = CountVectorizer()
feature_matrix = vectorizer.fit_transform(text_data)
feature_names = vectorizer.get_feature_names_out()
```

```
feature_names
```

```
array(['09062021', '09th', '10', ..., 'zx2', 'ßtill', 'ít'],
dtype=object)
```

```
import numpy as np

# Create a NumPy array of strings
arr = np.array(['10', '100', '1010', 'yr', 'zero', 'zip'],
dtype=object)

# Print the array
print(arr)
```

```
['10' '100' '1010' 'yr' 'zero' 'zip']
```

```
 import sklearn.feature_extraction.text as text
count_vectorizer = text.CountVectorizer()
```

```
count_vectorizer.fit(df.Review)
```

```
CountVectorizer()
```

```
data_features = count_vectorizer.transform(df.Review)
```

```
density = (data_features.getnnz() * 100) / (data_features.shape[0] *
data_features.shape[1])
print("Density of the matrix: ", density)
```

```
Density of the matrix:  0.4754105115684941
```

```
 feature_counts = df['Review'].value_counts()
feature_counts
```

```
Review
good
54
nice
33
nice product
29
very good
16
super
11
```

```
..
good tv in this price rangeaudio quality is average but in this price
everything is acceptable installation was also smooth and the
technician did it in time and was cooperative tooawesome smart tv
recommended for buying
1
super and high quality im so happy big display  and  super hd super
and high quality tv stand
1
reasonable price good performance safe and fast delivery allways a
pleasure shopping at flipkart thank you
1
hi guys this is an amazing itemprospicture excellent picture quality
for the price paid it is written hd ready but believe me you can play
1080p seamlessly audio volume at 20 is more than enough for a 15 x 10
feet room i tried upto 50 still 50 is left that kind of bomb boom
quality beats of music so nice at price pointconnectivity easy wifi
access no problem at all apps i didnt tried to install other apps yet
bcoz alot of usefull apps are preinstalled most of t     1
worth the money u spend for this monitor great deal using for cctv
footage monitorwonderful built msi brand which we can trust for
1
Name: count, Length: 1335, dtype: int64
```

```python
import numpy as np
import pandas as pd

# Assuming `vectorizer` and `data_features` are already defined

# Get feature names from the vectorizer
features = vectorizer.get_feature_names_out()

# Sum the counts of each feature across all samples
features_counts = np.sum(data_features.toarray(), axis=0)

# Create a DataFrame with features and their counts
features_counts_df = pd.DataFrame({'features': features, 'counts':
features_counts})

# Print or inspect the DataFrame
print(features_counts_df)
```

```
      features  counts
0     09062021       1
1         09th       3
2           10      49
3          100      27
4         1000       7
...        ...     ...
```

```
5388     zoom2       3
5389   zooming       2
5390       zx2       1
5391     ßtill       1
5392        ít       1

[5393 rows x 2 columns]
```

```
count_of_single_occurrences =
len(features_counts_df[features_counts_df['counts'] == 1])
count_of_single_occurrences
```

```
2060
```

```
count_vectorizer = CountVectorizer(max_features=10000)
feature_vector = count_vectorizer.fit_transform(df['Review'])
features = count_vectorizer.get_feature_names_out()
data_features = feature_vector.toarray()
features_counts = np.sum(data_features, axis=0)
feature_counts = pd.DataFrame({'features': features, 'counts':
features_counts})

top_features_counts = feature_counts.sort_values('counts',
ascending=False).head(15)

top_features_counts
```

```
{"summary":"{\n  \"name\": \"top_features_counts\",\n  \"rows\": 15,\n
\"fields\": [\n    {\n      \"column\": \"features\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 15,\n        \"samples\": [\n          \"in\",\
n          \"of\",\n          \"is\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"counts\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 685,\n
\"min\": 675,\n        \"max\": 3225,\n        \"num_unique_values\":
15,\n        \"samples\": [\n          1042,\n          771,\n
3225\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"top_features_counts"}
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
english_stop_words = stopwords.words('english')
```

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

```
 df['Review'][0:10]
```

```
0    best under 60k great performancei got it for a...
1                                    good perfomence
2    great performance but usually it has also that...
3             my wife is so happy and best product 😘
4    light weight laptop with new amazing features ...
5    amazing laptop am so much happy thanks for fli...
6              over all a good laptop for personal use
7                          thank you so much flipkart
8                                    amazing product
9    good for normal work  students online classes ...
Name: Review, dtype: object
```

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
X_train, X_test, y_train, y_test = train_test_split(df['Review'],
df['Rating'], test_size=0.2, random_state=42)
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
model = SVC()
model.fit(X_train_vectorized, y_train)
y_pred = model.predict(X_test_vectorized)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
5
print("Accuracy: ", accuracy)
print("Classification Report:\n", report)
```

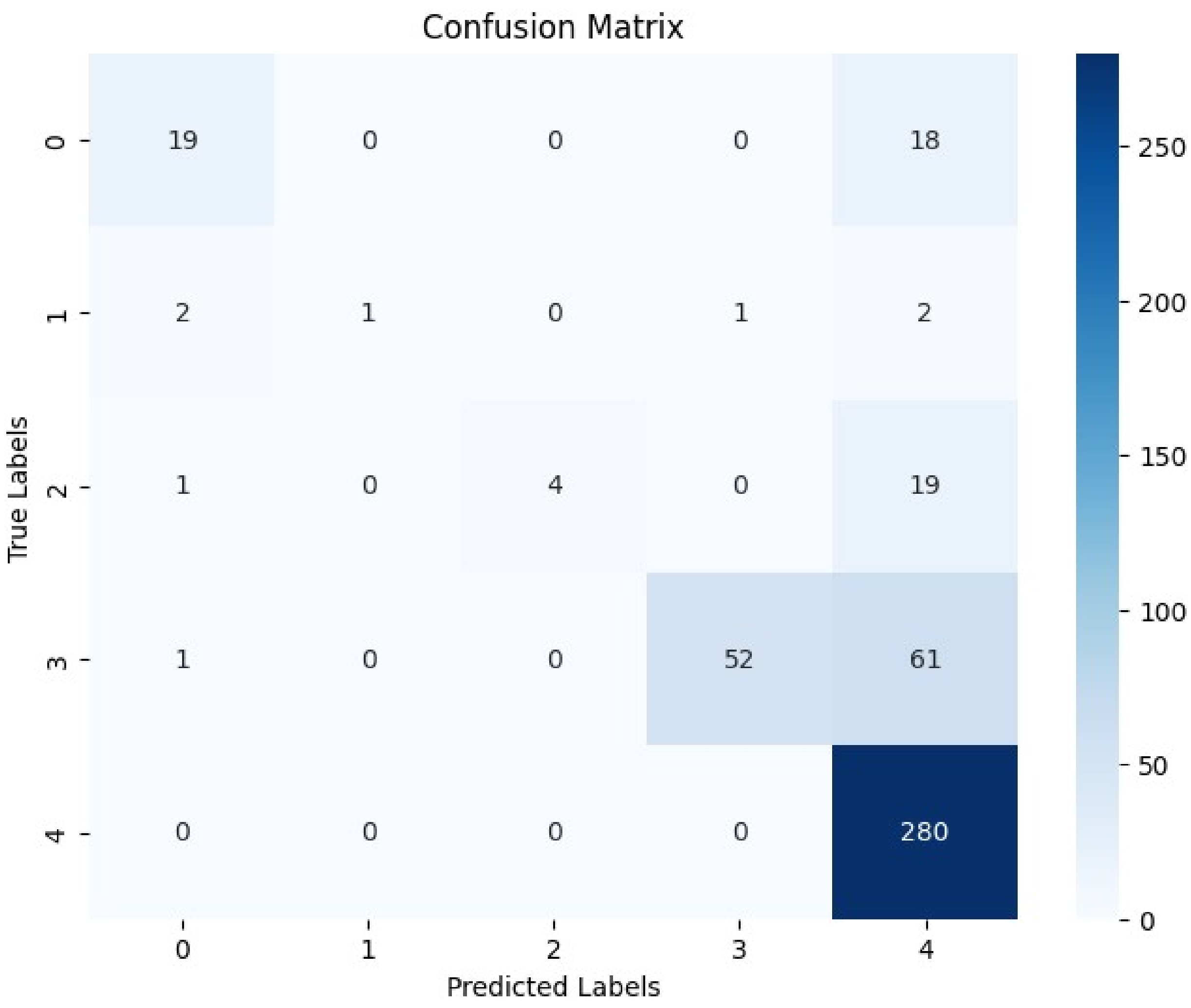
```
Accuracy:  0.7722342733188721
Classification Report:
               precision    recall  f1-score   support

           1       0.83      0.51      0.63        37
           2       1.00      0.17      0.29         6
           3       1.00      0.17      0.29        24
           4       0.98      0.46      0.62       114
           5       0.74      1.00      0.85       280

    accuracy                           0.77       461
   macro avg       0.91      0.46      0.54       461
weighted avg       0.82      0.77      0.74       461
```

```
print("Accuracy: ", accuracy)
print("Classification Report:\n", report)
```

```
Accuracy:  0.7722342733188721
Classification Report:
               precision    recall  f1-score   support
```

```
           1       0.83      0.51      0.63        37
           2       1.00      0.17      0.29         6
           3       1.00      0.17      0.29        24
           4       0.98      0.46      0.62       114
           5       0.74      1.00      0.85       280

    accuracy                           0.77       461
   macro avg       0.91      0.46      0.54       461
weighted avg       0.82      0.77      0.74       461
```

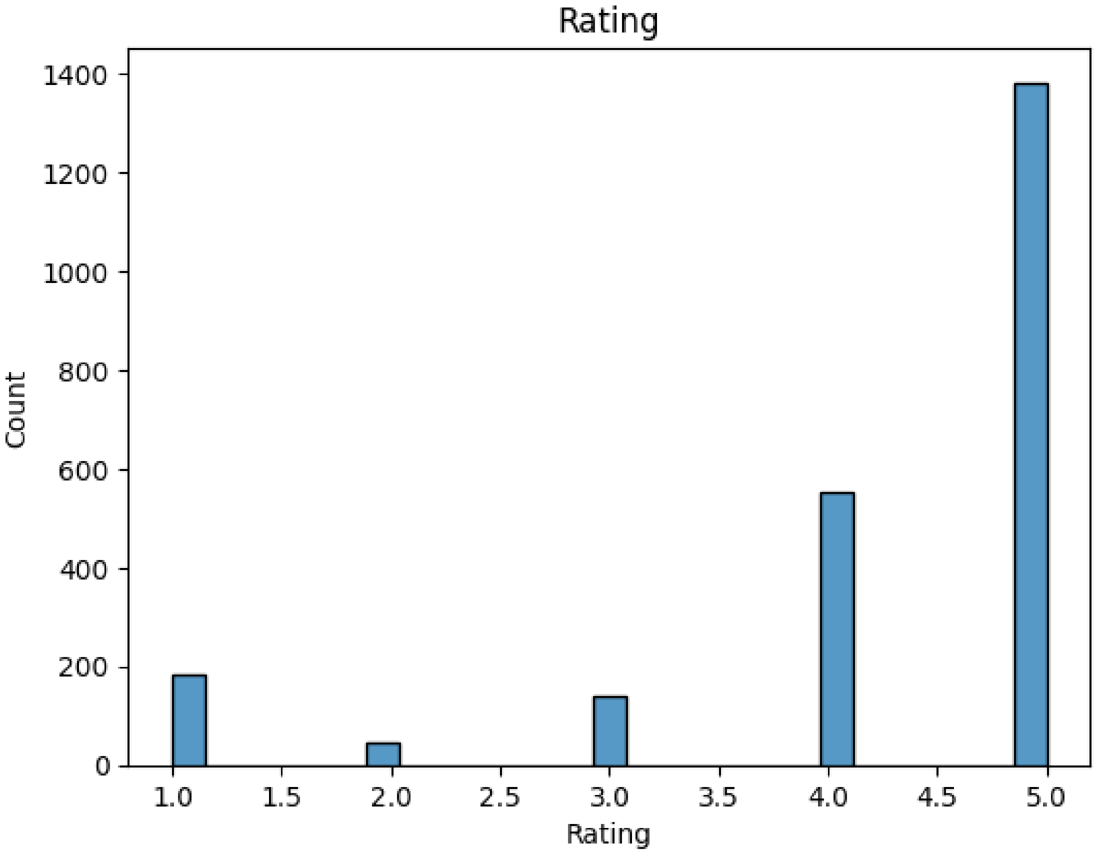
```
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



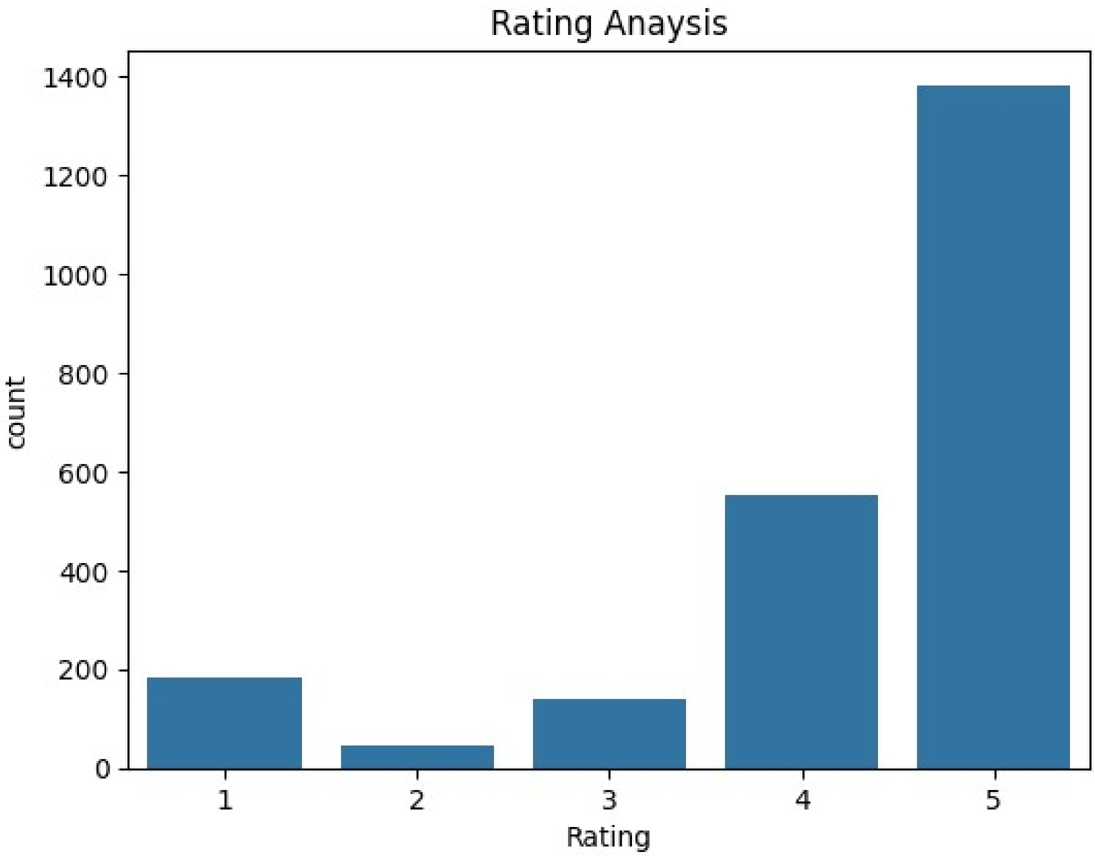
```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(df['Review'],
df['Rating'], test_size=0.2, random_state=42)
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
model = RandomForestClassifier()
model.fit(X_train_vectorized, y_train)
y_pred = model.predict(X_test_vectorized)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Accuracy: ", accuracy)
print("Classification Report:\n", report)
```

```
Accuracy:  0.8546637744034707
Classification Report:
               precision    recall  f1-score   support
```

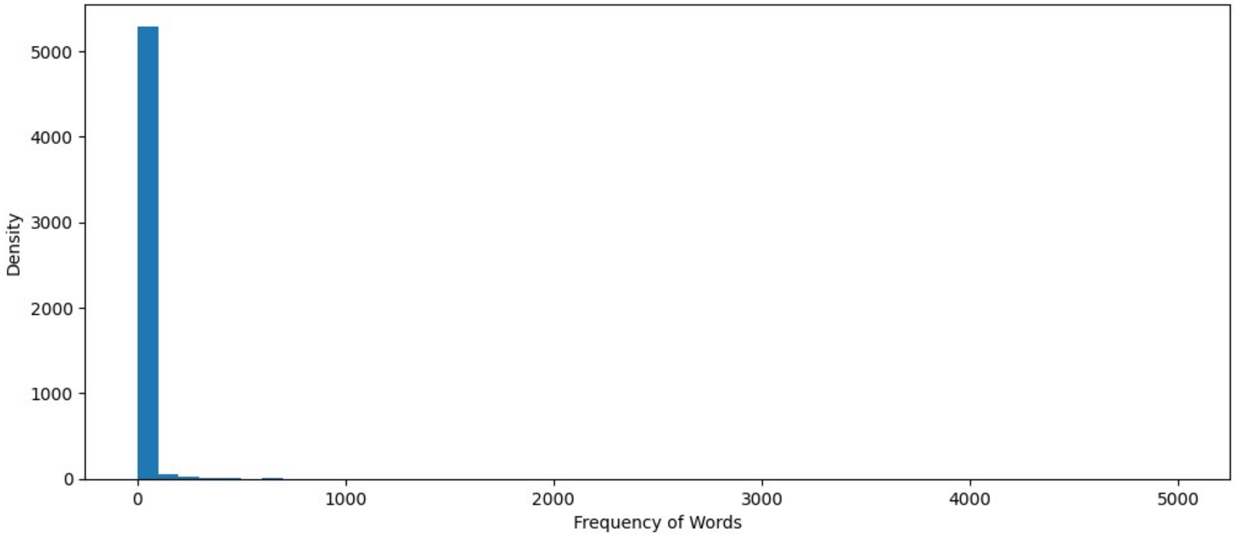
```
           1       1.00      0.89      0.94        37
           2       0.86      1.00      0.92         6
           3       0.93      0.54      0.68        24
           4       0.94      0.58      0.72       114
           5       0.82      0.99      0.89       280

    accuracy                           0.85       461
   macro avg       0.91      0.80      0.83       461
weighted avg       0.87      0.85      0.84       461


import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(df['Rating'])
plt.title('Rating')
plt.show()
```



```
sns.countplot(data=df, x='Rating')
plt.title('Rating Anaysis')
plt.show()
```



```python
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.hist(features_counts_df['counts'], bins=50, range=(0, 5000))
plt.xlabel('Frequency of Words')
plt.ylabel('Density')
plt.show()
```


Density
Frequency of Words