```
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/amazon_reviews.csv")

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3150,\n  \"fields\":
[\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      \"column\": \"date\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 77,\n
\"samples\": [\n          \"27-Jul-18\",\n          \"26-Jun-18\",\n
\"21-Jul-18\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"variation\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 16,\n
\"samples\": [\n          \"Charcoal Fabric \",\n          \"Walnut
Finish \",\n          \"Black\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"verified_reviews\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 2300,\n        \"samples\": [\n          \"Fun
tech toy\",\n          \"Love the fire stick. Alexa works well on it
too. Would recommend.\",\n          \"The best part of this product is
you can control the thermostat and lights for your house.  There isn\\
u2019t anything I dislike.\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"feedback\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

```
 0   rating            3150 non-null   int64
 1   date              3150 non-null   object
 2   variation         3150 non-null   object
 3   verified_reviews  3149 non-null   object
 4   feedback          3150 non-null   int64
dtypes: int64(2), object(3)
memory usage: 123.2+ KB
```

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

```
Null values in the entire Data:
rating              0
date                0
variation           0
verified_reviews    1
feedback            0
dtype: int64
```

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

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

```
rating              0
date                0
variation           0
verified_reviews    0
feedback            0
dtype: int64
```

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

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

```
df['verified_reviews']
```

```
0                                                Love my Echo
1                                                    Loved it
2       Sometimes while playing a game you can answer ...
3       I have had a lot of fun with this thing My 4 y...
4                                                       Music
                              ...
2796    I do love these things i have them running my ...
2797    Only complaint I have is that the sound qualit...
2798                                                    Good
```

```
2799                           Nice little unit  no issues
2800    The echo dot was easy to set up and use It hel...
Name: verified_reviews, Length: 2434, dtype: object
```

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

```
feature_names
```

```
array(['072318', '10', '100', ..., 'zzzz', 'zzzzzzz', 'útil'],
      dtype=object)
```

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

```
count_vectorizer.fit(df.verified_reviews)
```

```
CountVectorizer()
```

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

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

```
Density of the matrix:  0.44711257922647296
```

```
feature_counts = df['verified_reviews'].value_counts()
feature_counts # Display the calculated feature counts
```

```
verified_reviews

50
Love it
31
I love it
12
Great product
12
Works great
10


..
We love our echo spots We now have three through the house and like
that you can drop in on the other spots We do wish the clock faces had
more customization options and that there were more faces in general
1
Like the little spot  now our guest room is Alexa enabled
```

```
1
This is our first step into a smart home Soon as I added one item to the grocery list I started loving it My wife was soso about it ‘til I opened rain sounds She’s a convert ☺ LOVE these and the dots
1
It won’t receive calls and it’s really hard to call someone who has another echo
1
The echo dot was easy to set up and use It helps provide music etc to small spaces and was just what I was looking for
1
Name: count, Length: 2260, dtype: int64
```

```
features = vectorizer.get_feature_names_out()
# Replace with the variable that holds feature names
features_counts = np.sum(data_features.toarray(), axis=0)
features_counts_df = pd.DataFrame({'features': features,
'counts':features_counts})

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

```
2310
```

```
count_vectorizer = CountVectorizer(max_features=10000)
feature_vector =
count_vectorizer.fit_transform(df['verified_reviews'])
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
\"echo\",\n          \"of\",\n          \"the\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"counts\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 712,\n
\"min\": 544,\n        \"max\": 2653,\n        \"num_unique_values\":
15,\n        \"samples\": [\n          652,\n          623,\n
2653\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['verified_reviews'][0:10]
```

```
0                                          Love my Echo
1                                               Loved it
2    Sometimes while playing a game you can answer ...
3    I have had a lot of fun with this thing My 4 y...
4                                                 Music
5    I received the echo as a gift I needed another...
6    Without having a cellphone I cannot use many o...
7    I think this is the 5th one Ive purchased Im w...
8                                           looks great
9    Love it I’ve listened to songs I haven’t heard...
Name: verified_reviews, dtype: object
```

```
# Verify if 'Sentiment' column exists. If not, create it based on your
problem
if 'Sentiment' not in df.columns:
    # Example: Create 'Sentiment' based on ratings (adjust logic as
needed)
    df['Sentiment'] = df['verified_reviews'].apply(lambda rating:
'positive' if rating > 3 else 'negative')

# Proceed with your train_test_split and model training
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['verified_reviews'],df['Sentiment'],
test_size=0.2, random_state=42)
# ... rest of your code ...

import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer # Import
TfidfVectorizer
```

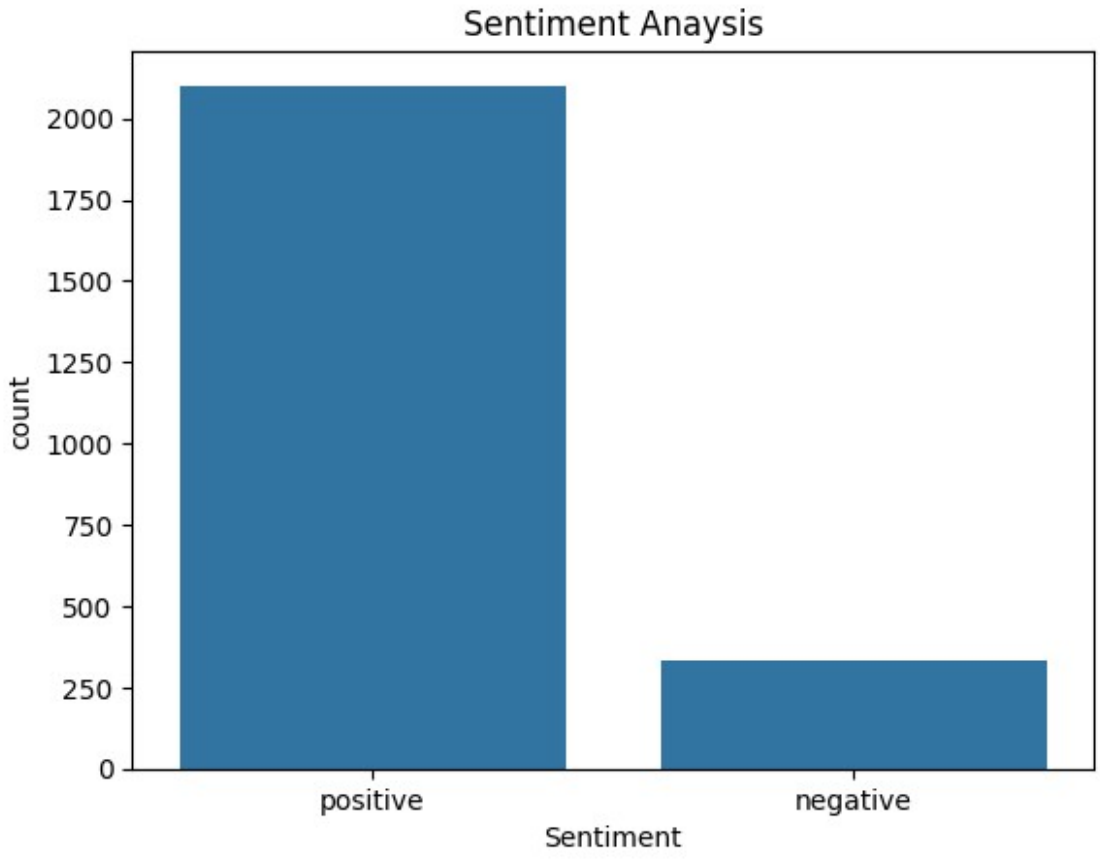
```
# Assuming df, X_train, X_test, y_train, y_test are already defined
# Preprocess text data using TF-IDF vectorization
vectorizer = TfidfVectorizer()  # Initialize TfidfVectorizer
X_train = vectorizer.fit_transform(X_train)

from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test =
train_test_split(df['verified_reviews'],df['Sentiment'],
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:  0.8542094455852156
Classification Report:
              precision    recall  f1-score   support

    negative       0.82      0.12      0.20        78
    positive       0.86      1.00      0.92       409

    accuracy                           0.85       487
   macro avg       0.84      0.56      0.56       487
weighted avg       0.85      0.85      0.80       487
```

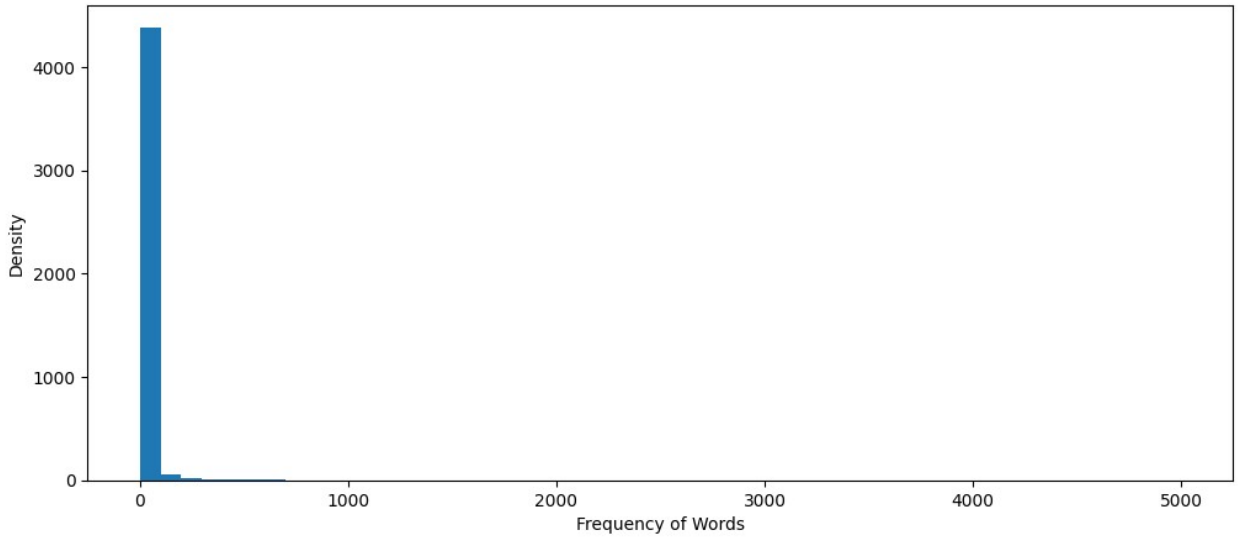
```
import matplotlib.pyplot as plt
import seaborn as sns

# Check the available columns in your DataFrame
print(df.columns)

# If 'product_price' is not present, handle the situation accordingly:
# 1. Correct the column name if it's a typo.
# 2. If the column is missing, you might need to load or process the
data to include it.

# Example (assuming 'product_price' is the correct column name):
if 'product_price' in df.columns:
    sns.histplot(df['product_price'])
    plt.title('Product Price')
    plt.show()
else:
    print("Column 'product_price' not found in the DataFrame.")
```

```
Index(['rating', 'date', 'variation', 'verified_reviews', 'feedback',
       'verified_reviews\t', 'Sentiment'],
      dtype='object')
Column 'product_price' not found in the DataFrame.
```

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




```
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()
```


4000
3000
2000
1000
0
Density
0
1000
2000
3000
4000
5000
Frequency of Words