sentimental

June 28, 2024

[ ]:
```
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
```

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[7]:
```
import pandas as pd # Import the pandas library and give it the alias 'pd'

df = pd.read_csv("/content/drive/MyDrive/Twitter And Reddit/Reddit_Data.csv")
```

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

[8]:

	clean_comment	category
0	family mormon have never tried explain them t…	1
1	buddhism has very much lot compatible with chr…	1
2	seriously don say thing first all they won get…	-1
3	what you have learned yours and only yours wha…	0
4	for your own benefit you may want read living …	1

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37249 entries, 0 to 37248
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   clean_comment  37149 non-null  object
 1   category       37249 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 582.1+ KB
```

[10]:
```
null_values = df.isnull().sum()
print("Null values in the entire Data:")
print(null_values)
```

```
Null values in the entire Data:
clean_comment    100
category           0
dtype: int64
```

[11]:
```
df.dropna(inplace=True)
```

[12]:
```
null_values = df.isnull().sum()
null_values
```

[12]:
```
clean_comment    0
category         0
dtype: int64
```

[13]:
```
df.drop_duplicates(inplace=True)
```

[15]:
```
import string  # Import the string module

# Check if 'Review' column exists, if not, try a different column name or fix␣
 ↪the previous steps
if 'clean_comment' in df.columns:
    df['clean_comment'] = df['clean_comment'].apply(lambda x: x.lower())
    df['clean_comment'] = df['clean_comment'].apply(lambda x: x.translate(str.
 ↪maketrans('', '',string.punctuation))) # Now you can use string.punctuation
```

[16]:
```
df['clean_comment']
```

[16]:
```
0         family mormon have never tried explain them t…
1        buddhism has very much lot compatible with chr…
2        seriously don say thing first all they won get…
3        what you have learned yours and only yours wha…
4        for your own benefit you may want read living …
                              …
37244                                              jesus
37245    kya bhai pure saal chutiya banaya modi aur jab…
37246                downvote karna tha par upvote hogaya
37247                                          haha nice
37248               facebook itself now working bjp' cell
Name: clean_comment, Length: 36799, dtype: object
```

[17]:
```
from sklearn.feature_extraction.text import CountVectorizer
# Assuming 'df' is your Data containing text data
text_data = df['clean_comment']
```

```
vectorizer = CountVectorizer()
feature_matrix = vectorizer.fit_transform(text_data)
feature_names = vectorizer.get_feature_names_out()
```

[18]:
```
feature_names
```

[18]:
```
array(['000', '0001', '000cr', …, '      ', '  ', '  '],
      dtype=object)
```

[19]:
```
import sklearn.feature_extraction.text as text
count_vectorizer = text.CountVectorizer()
```

[20]:
```
count_vectorizer.fit(df.clean_comment)
```

[20]:
```
CountVectorizer()
```

[21]:
```
data_features = count_vectorizer.transform(df.clean_comment)
```

[22]:
```
density = (data_features.getnnz() * 100) / (data_features.shape[0] *
data_features.shape[1])
print("Density of the matrix: ", density)
```

```
Density of the matrix:  0.04316026154575147
```

[23]:
```
feature_counts = df['clean_comment'].value_counts()
feature_counts
```

[23]:
```
clean_comment
                       3
 vreddit bot
2
                       2
just read article opindia talking about how the report itself fake news haven
laughed much while opindia saying that bbc fake news
1
hey there are lot whatsapp forward like this and the election are coming there
will more whatsapp forward wrong facts and hatred for every forward neither you
nor anyone other write fact basis answer like this can aggregate all the
forwards one place and assign best factual answer that whenever receive forward
don have type whole message just can visit the aggregated place website and pick
answer for forward and reply that anyone can contribute this and doing this will
aware what being forward     1
                      ..
how many people here remember time before fox news
1
waiting for result and lets see who will win and make govt punjab
1
```

```
 you see split secession civil war the horizon
1
what the general consensus legacy admission the top universities why don more
people push remove them instead debating affirmative action
1
facebook itself now working bjp' cell
1
Name: count, Length: 36795, dtype: int64
```

[26]:
```
import numpy as np # Import the NumPy library

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

[27]:
```
count_of_single_occurrences␣
 ↪=len(features_counts_df[features_counts_df['counts'] == 1])
count_of_single_occurrences
```

[27]:
```
26173
```

[28]:
```
count_vectorizer = CountVectorizer(max_features=10000)
feature_vector = count_vectorizer.fit_transform(df['clean_comment'])
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})
```

[29]:
```
top_features_counts = feature_counts.sort_values('counts', ascending=False).
 ↪head(15)
```

[30]:
```
top_features_counts
```

[30]:

	features	counts
8987	the	57717
501	and	28957
8985	that	15455
9022	this	13475
3521	for	12982
9965	you	11794
628	are	10567
9009	they	8708
6097	not	8695
4030	have	8382

```
9853     with    7845
1389      but    7271
9824     will    6906
9713      was    6373
6535   people    5557
```

[31]:
```
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.
```

[32]:
```
df['clean_comment'][0:10]
```

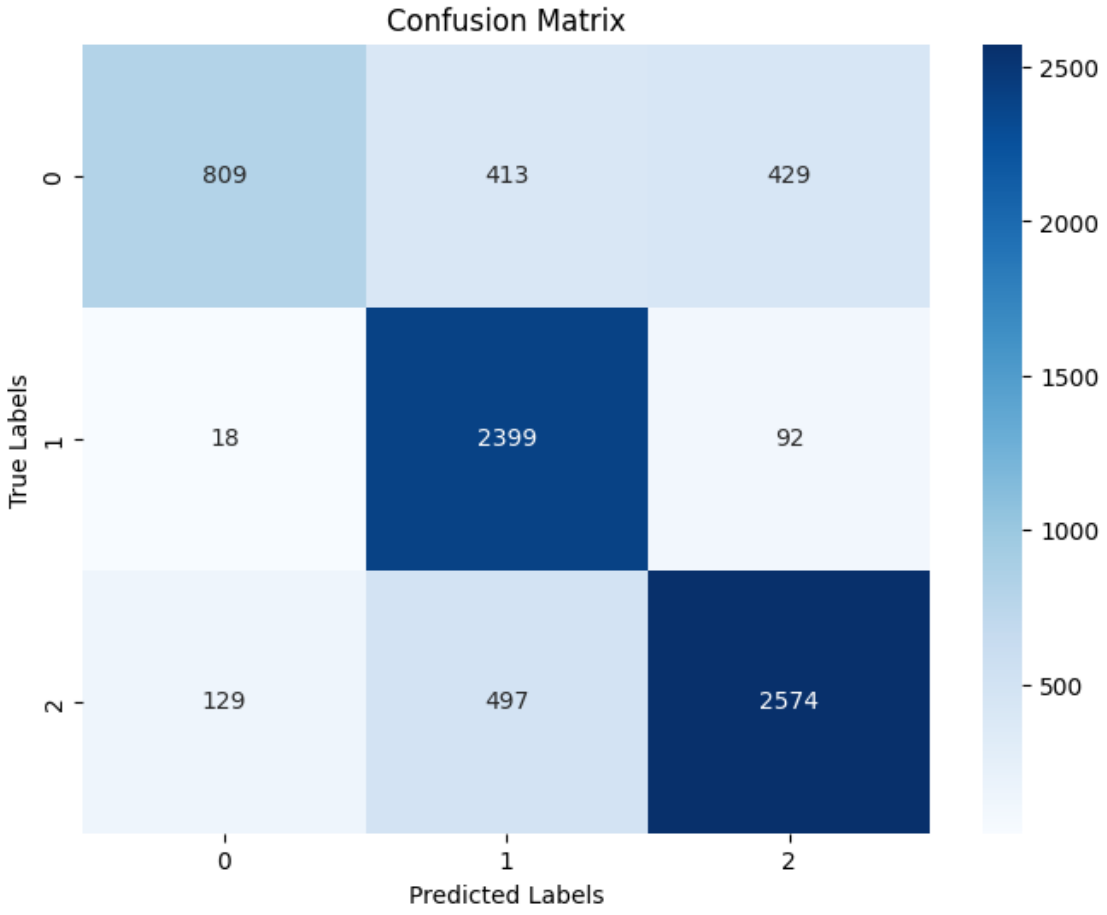
[32]:
```
0     family mormon have never tried explain them t…
1    buddhism has very much lot compatible with chr…
2    seriously don say thing first all they won get…
3    what you have learned yours and only yours wha…
4    for your own benefit you may want read living …
5    you should all sit down together and watch the…
6     was teens when discovered zen meditation was …
7                              jesus was zen meets jew
8    there are two varieties christians dogmatic th…
9    dont worry about trying explain yourself just …
Name: clean_comment, dtype: object
```

[33]:
```
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['clean_comment'],df['category'], 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)
print("Accuracy: ", accuracy)
print("Classification Report:\n", report)
```

```
Accuracy:  0.7855978260869565
Classification Report:
```

	precision	recall	f1-score	support
-1	0.85	0.49	0.62	1651
0	0.72	0.96	0.82	2509
1	0.83	0.80	0.82	3200
accuracy			0.79	7360
macro avg	0.80	0.75	0.75	7360
weighted avg	0.80	0.79	0.78	7360

[34]:
```
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()
```

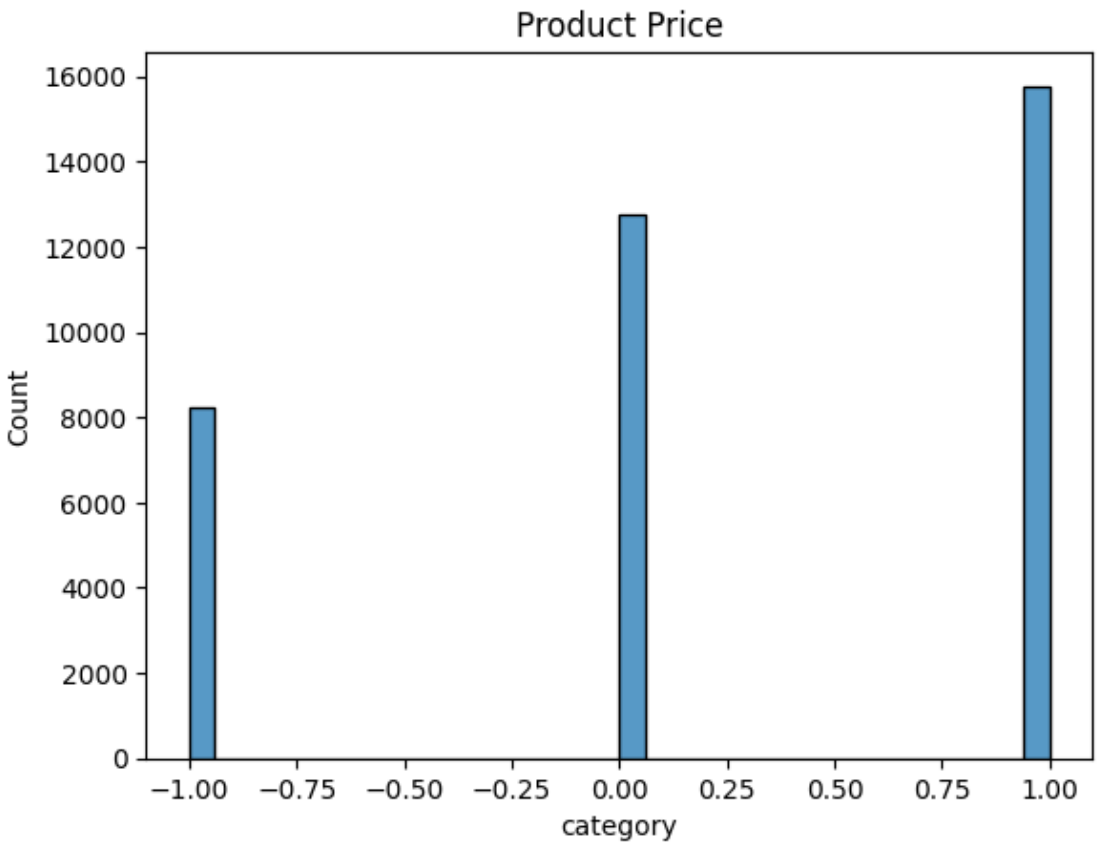


[38]:
```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(df['clean_comment'],df['category'], test_size=0.2,␣
 ↪random_state=42)
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
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)
```

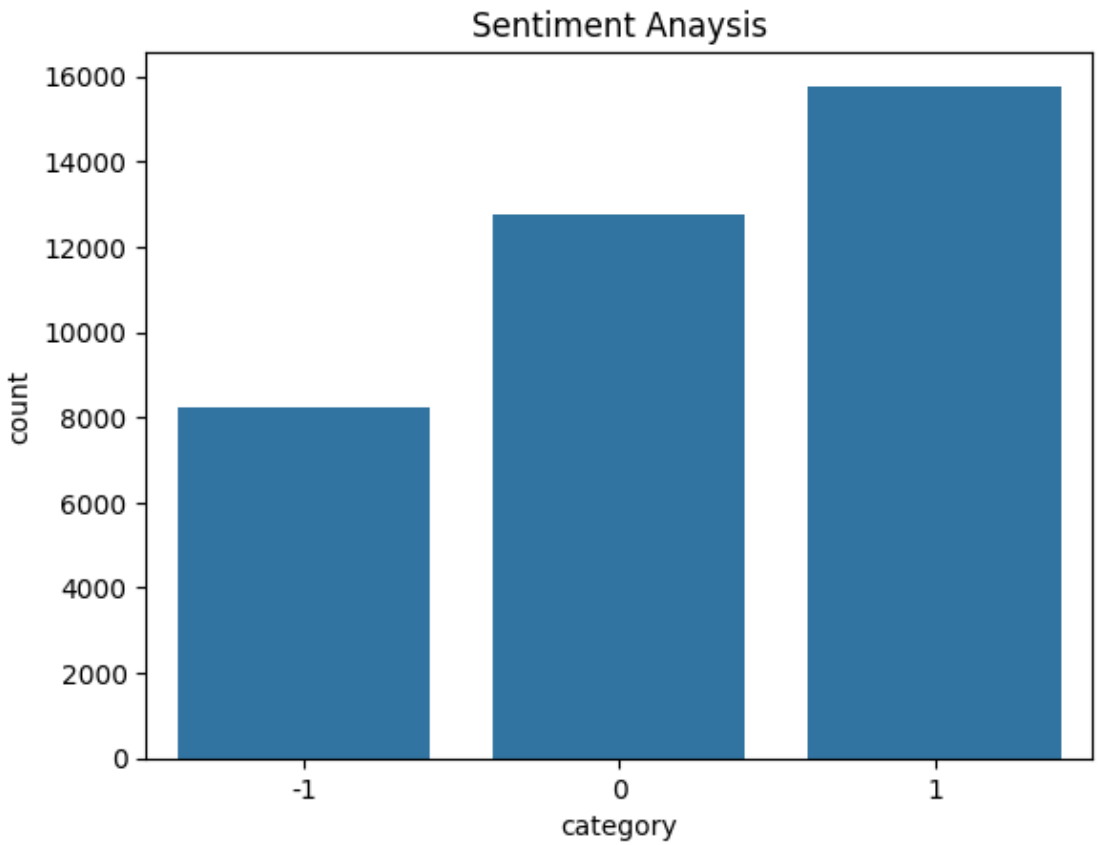
```
Accuracy:  0.7817934782608695
Classification Report:
              precision    recall  f1-score   support

          -1       0.92      0.33      0.49      1651
           0       0.79      0.94      0.86      2509
           1       0.75      0.89      0.82      3200

    accuracy                           0.78      7360
   macro avg       0.82      0.72      0.72      7360
weighted avg       0.80      0.78      0.76      7360
```

[42]:
```
import matplotlib.pyplot as plt
import seaborn as sns

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

# If you have a column representing price, replace 'actual_price_column' with␣
 ↪the correct column name
sns.histplot(df['category'])

plt.title('Product Price')
plt.show()
```

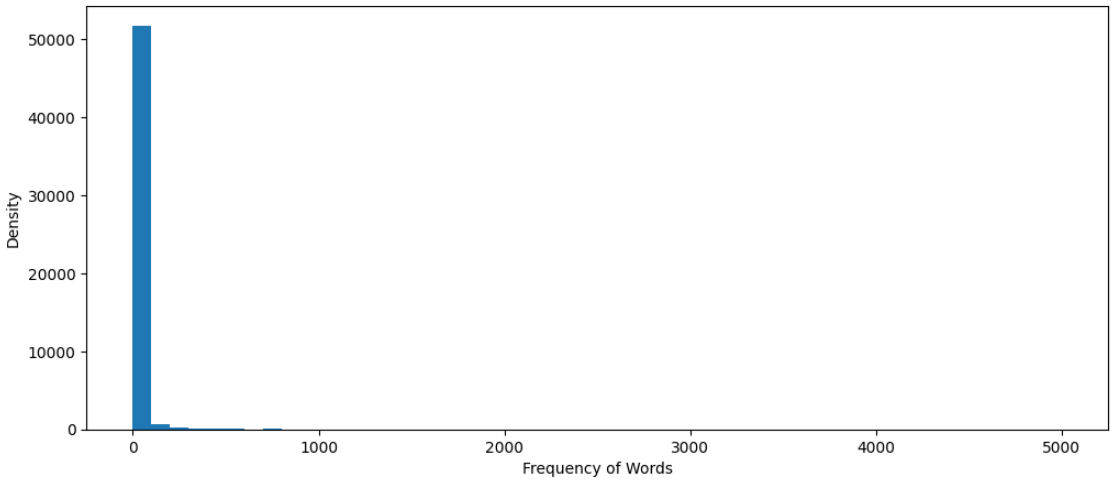
```
Index(['clean_comment', 'category'], dtype='object')
```



[45]:
```
sns.countplot(data=df, x='category')
plt.title('Sentiment Anaysis')
plt.show()
```



[46]:
```
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()
```



[ ]: