sentimental

June 28, 2024

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

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

[ ]:
```
import pandas as pd
import numpy as nm
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import re
import string
import matplotlib.pyplot as plt
```

[ ]:
```
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/yt_oldest.csv") # Replace 'your_file.
↪csv' with the actual name of your CSV file
```

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

[ ]:
```
                  title     video_id                channel_id  \
0           Me at the zoo  jNQXAC9IVRw  UC4QobU6STFB0P71PMvOGN5A
1   My Snowboarding Skillz          NaN  UC7DazZtuimjEKKjU2M6aq8w
2                tribute          NaN  UCBE8t4E44we_imhhyszl1UA
3     carrie rides a truck          NaN  UC7K5am1UAQEsCRhzXpi9i1g
4          Vernal Lullaby          NaN  UC1QYeQWfUg0zSyMPPCagpXQ

  channel_title          published_at  view_count  like_count  comment_count
0         jawed  2005-04-24T03:31:52Z   302611983    15514886       10376746
1           NaN  2005-04-24T03:56:09Z     3690565      164284          29119
2           NaN  2005-04-24T22:15:20Z     2556256       79806           9832
3           NaN  2005-04-30T14:24:14Z      512381        6748           2064
```

```
4            NaN  2005-05-03T02:33:33Z      440455         8829            2378
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 964 entries, 0 to 963
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   title          964 non-null    object
 1   video_id       957 non-null    object
 2   channel_id     401 non-null    object
 3   channel_title  393 non-null    object
 4   published_at   964 non-null    object
 5   view_count     964 non-null    int64
 6   like_count     964 non-null    int64
 7   comment_count  964 non-null    int64
dtypes: int64(3), object(5)
memory usage: 60.4+ KB
```

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

```
Null values in the entire Data:
title              0
video_id           7
channel_id       563
channel_title    571
published_at       0
view_count         0
like_count         0
comment_count      0
dtype: int64
```

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

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

```
[ ]: title            0
     video_id         0
     channel_id       0
     channel_title    0
     published_at     0
     view_count       0
     like_count       0
```

```
comment_count    0
dtype: int64
```

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

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

```
[ ]: df['title']
```

```
[ ]: 0                                           me at the zoo
     5                                              good times
     9                                                   gtasa
     10                            wanna see brian very happy
     11                                         lots a paypa
                                          …
     396    take me to your heart  mltr  hyesung korean ve…
     397                    audioslave  i am the highway live
     398                        glynis  the smashing pumpkins
     399                                alexz johnson24 hours
     400                               code lyoko xana attack
     Name: title, Length: 363, dtype: object
```

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

```
[ ]: feature_names
```

```
[ ]: array(['001', '02', '05', …, '   ', '   ', '      '],
           dtype=object)
```

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

```
[ ]: count_vectorizer.fit(df.title)
```

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

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

```
Density of the matrix:  0.38672438672438675
```

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

```
[ ]: title
     fight                                            3
     lebron james powerade commercial                 1
     liz mcclarnon woman in love music video          1
     opening to ah my goddess                         1
     qoo  snow                                        1
                                                     ..
     los pedos si son flamables                       1
     argentina vs colombia  1995  gol de crespo  mdq  1
     fushigi yuugi  pretty girl                       1
     fiddy fun                                        1
     code lyoko xana attack                           1
     Name: count, Length: 361, 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
```

```
[ ]: 883
```

```
[ ]: count_vectorizer = CountVectorizer(max_features=10000)
     feature_vector = count_vectorizer.fit_transform(df['title'])
     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
[ ]:      features  counts
    893        the      59
    651         of      18
    957      video      17
    902         to      14
    612      music      13
    657         on      13
    1012       you      12
    578         me      11
    510         la       9
    914    trailer       9
    443         in       8
    59         and       8
    995       with       7
    534       live       7
    234      dance       7
```

```
[ ]: 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['title'][0:10]
```

```
[ ]: 0                                        me at the zoo
     5                                           good times
     9                                                gtasa
     10                          wanna see brian very happy
     11                                     lots a paypa
     12    la signora franca e la pacifica contestazione
     13                                      chocolate milk
     14                                       davood khatar
     15                                      carrol blunder
     16                                          chopsticks
     Name: title, dtype: object
```

```
[ ]: # Assuming you have a list named "sentiments" with the sentiment labels
     # Replace this with the actual way to get your sentiment labels
     # Example: Assuming you have a function called "get_sentiment" that returns the␣
      ↪sentiment for a given text
     def get_sentiment(text): # Define the get_sentiment function
         # Replace with your actual sentiment analysis logic here
         if 'good' in text:
             return 'positive'
```

```
    elif 'bad' in text:
        return 'negative'
    else:
        return 'neutral'

sentiments = [get_sentiment(text) for text in df['title']] # Get sentiments for␣
 ↪each title
df['Sentiment'] = sentiments # Add the 'Sentiment' column to your DataFrame
```

```
[ ]: 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 CountVectorizer

# Verify that 'Sentiment' column exists and correct if needed.
print(df.columns) # Print columns to check if 'Sentiment' exists

# If 'Sentiment' column does not exist, you need to create it or load it.
# For example, if you have sentiment labels in a list called 'sentiments':
# df['Sentiment'] = sentiments

# Assuming you have a list named "sentiments" with the sentiment labels
# Replace this with the actual way to get your sentiment labels
df['Sentiment'] = sentiments # Add the 'Sentiment' column to your DataFrame

X_train, X_test, y_train, y_test = train_test_split(df['title'],
df['Sentiment'], test_size=0.2, random_state=42) # Now this line should work
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)
```

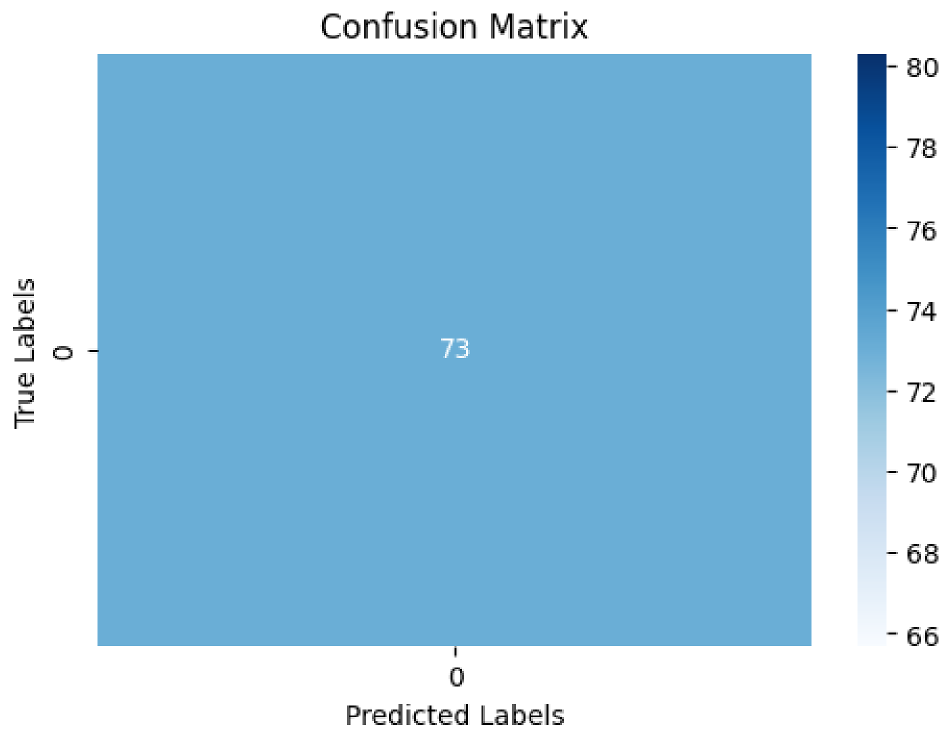
```
Index(['title', 'video_id', 'channel_id', 'channel_title', 'published_at',
       'view_count', 'like_count', 'comment_count', 'Sentiment'],
      dtype='object')
```

```
[ ]: 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['title'],
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 = 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:  1.0
Classification Report:
               precision    recall  f1-score   support

     neutral       1.00      1.00      1.00        73

    accuracy                           1.00        73
   macro avg       1.00      1.00      1.00        73
weighted avg       1.00      1.00      1.00        73
```

[ ]:
```
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=(6, 4))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



[85]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer

X_train, X_test, y_train, y_test = train_test_split(df['title'],␣
  ↪df['like_count'], 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:  0.0136986301369863
Classification Report:
               precision    recall  f1-score   support

           0       0.04      1.00      0.07         1
           3       0.00      0.00      0.00         1
```

7	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
10	0.00	0.00	0.00	1
30	0.00	0.00	0.00	0
58	0.00	0.00	0.00	1
79	0.00	0.00	0.00	1
97	0.00	0.00	0.00	0
101	0.00	0.00	0.00	0
103	0.00	0.00	0.00	0
113	0.00	0.00	0.00	1
127	0.00	0.00	0.00	0
166	0.00	0.00	0.00	1
180	0.00	0.00	0.00	1
216	0.00	0.00	0.00	1
226	0.00	0.00	0.00	1
235	0.00	0.00	0.00	1
236	0.00	0.00	0.00	0
245	0.00	0.00	0.00	1
292	0.00	0.00	0.00	1
342	0.00	0.00	0.00	0
384	0.00	0.00	0.00	0
414	0.00	0.00	0.00	1
445	0.00	0.00	0.00	1
458	0.00	0.00	0.00	1
503	0.00	0.00	0.00	0
527	0.00	0.00	0.00	1
538	0.00	0.00	0.00	1
541	0.00	0.00	0.00	1
659	0.00	0.00	0.00	1
691	0.00	0.00	0.00	0
693	0.00	0.00	0.00	1
745	0.00	0.00	0.00	1
755	0.00	0.00	0.00	0
769	0.00	0.00	0.00	1
801	0.00	0.00	0.00	0
813	0.00	0.00	0.00	1
855	0.00	0.00	0.00	1
888	0.00	0.00	0.00	1
902	0.00	0.00	0.00	0
922	0.00	0.00	0.00	1
928	0.00	0.00	0.00	1
943	0.00	0.00	0.00	1
997	0.00	0.00	0.00	1
1000	0.00	0.00	0.00	0
1017	0.00	0.00	0.00	0
1038	0.00	0.00	0.00	0
1113	0.00	0.00	0.00	1
1154	0.00	0.00	0.00	1

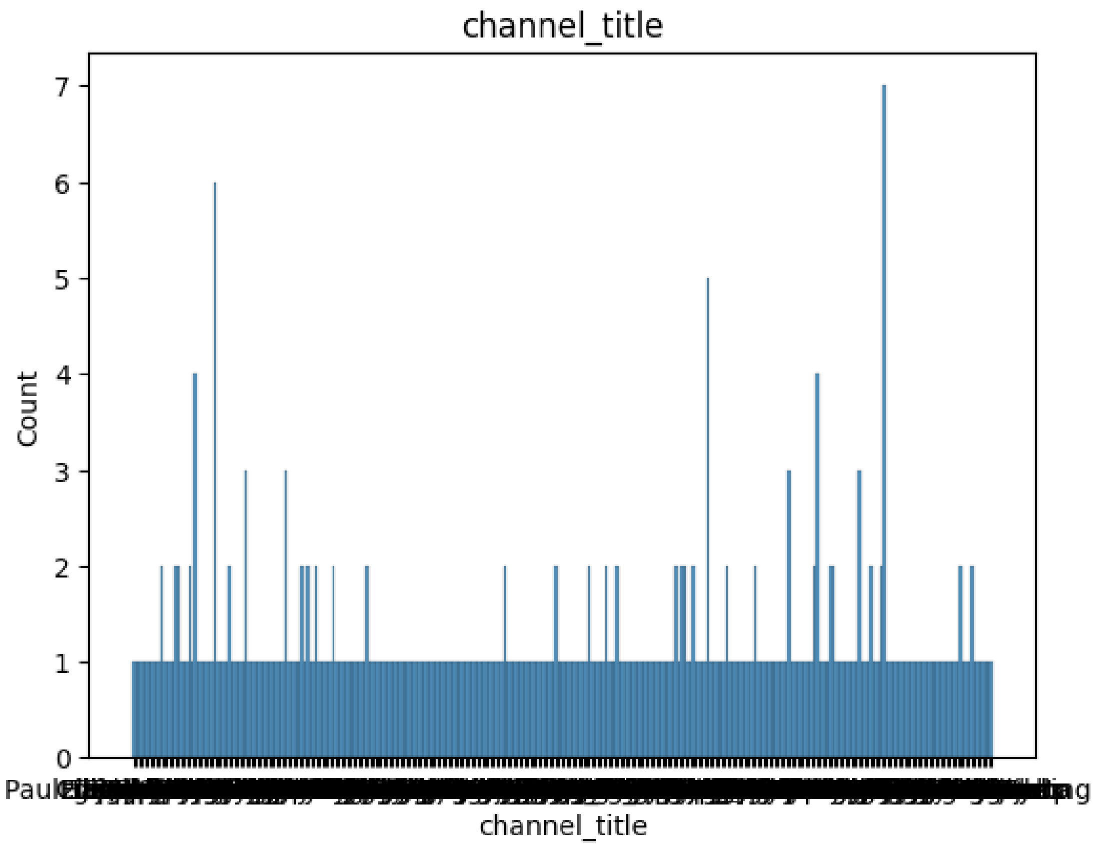
1161	0.00	0.00	0.00	1
1191	0.00	0.00	0.00	0
1206	0.00	0.00	0.00	1
1223	0.00	0.00	0.00	1
1257	0.00	0.00	0.00	1
1316	0.00	0.00	0.00	1
1322	0.00	0.00	0.00	0
1331	0.00	0.00	0.00	1
1347	0.00	0.00	0.00	1
1477	0.00	0.00	0.00	1
1543	0.00	0.00	0.00	0
1650	0.00	0.00	0.00	0
1757	0.00	0.00	0.00	0
1758	0.00	0.00	0.00	1
1902	0.00	0.00	0.00	1
1949	0.00	0.00	0.00	0
2150	0.00	0.00	0.00	0
2152	0.00	0.00	0.00	0
2452	0.00	0.00	0.00	1
2535	0.00	0.00	0.00	1
2578	0.00	0.00	0.00	1
2593	0.00	0.00	0.00	1
2689	0.00	0.00	0.00	1
2722	0.00	0.00	0.00	0
2868	0.00	0.00	0.00	0
3558	0.00	0.00	0.00	1
3652	0.00	0.00	0.00	1
4204	0.00	0.00	0.00	1
5018	0.00	0.00	0.00	0
5203	0.00	0.00	0.00	1
5454	0.00	0.00	0.00	1
5656	0.00	0.00	0.00	0
6363	0.00	0.00	0.00	1
7434	0.00	0.00	0.00	1
9310	0.00	0.00	0.00	1
10534	0.00	0.00	0.00	0
11978	0.00	0.00	0.00	1
12378	0.00	0.00	0.00	1
12637	0.00	0.00	0.00	1
13563	0.00	0.00	0.00	1
13759	0.00	0.00	0.00	1
16991	0.00	0.00	0.00	1
17949	0.00	0.00	0.00	1
24340	0.00	0.00	0.00	1
27214	0.00	0.00	0.00	1
31225	0.00	0.00	0.00	0
35194	0.00	0.00	0.00	0
54112	0.00	0.00	0.00	1

```
       63164       0.00      0.00      0.00         0
       64864       0.00      0.00      0.00         0
       67112       0.00      0.00      0.00         1
       79062       0.00      0.00      0.00         1
       79391       0.00      0.00      0.00         1
       91574       0.00      0.00      0.00         1
      551496       0.00      0.00      0.00         1
      576434       0.00      0.00      0.00         1

    accuracy                           0.01        73
   macro avg       0.00      0.01      0.00        73
weighted avg       0.00      0.01      0.00        73

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

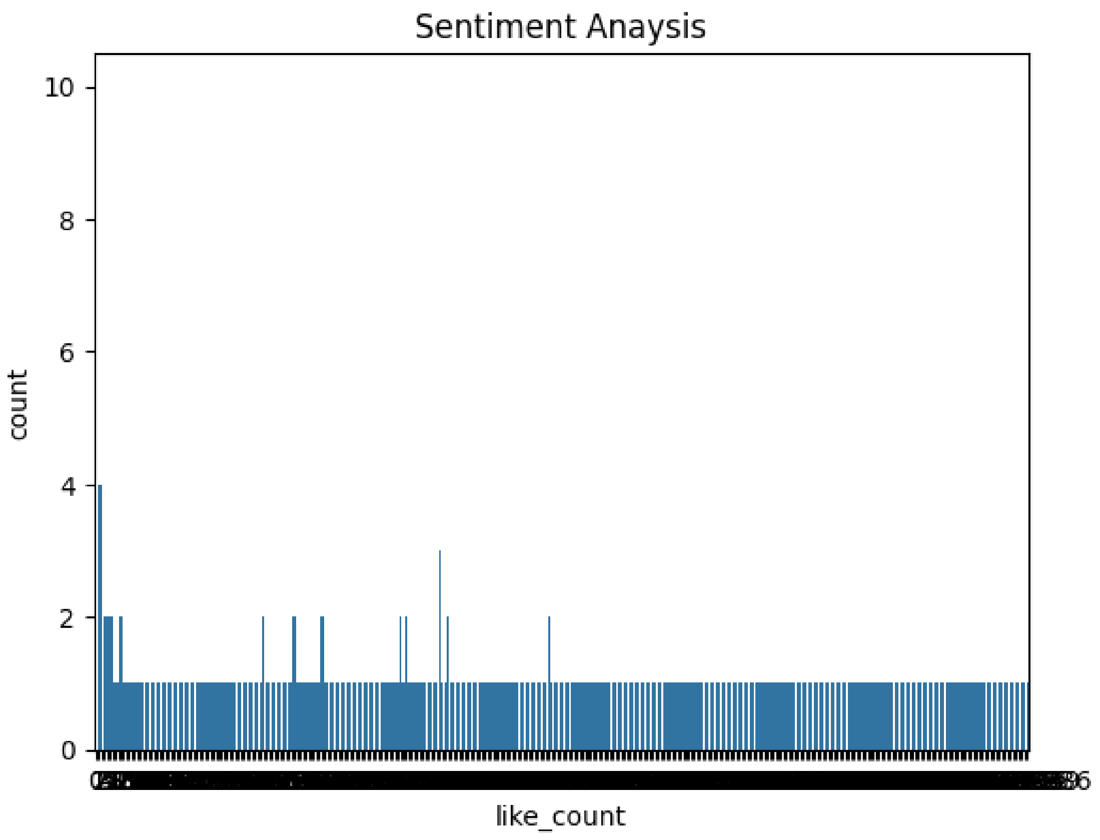
[80]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(df['channel_title'])
```

```
plt.title('channel_title')
plt.show()
```

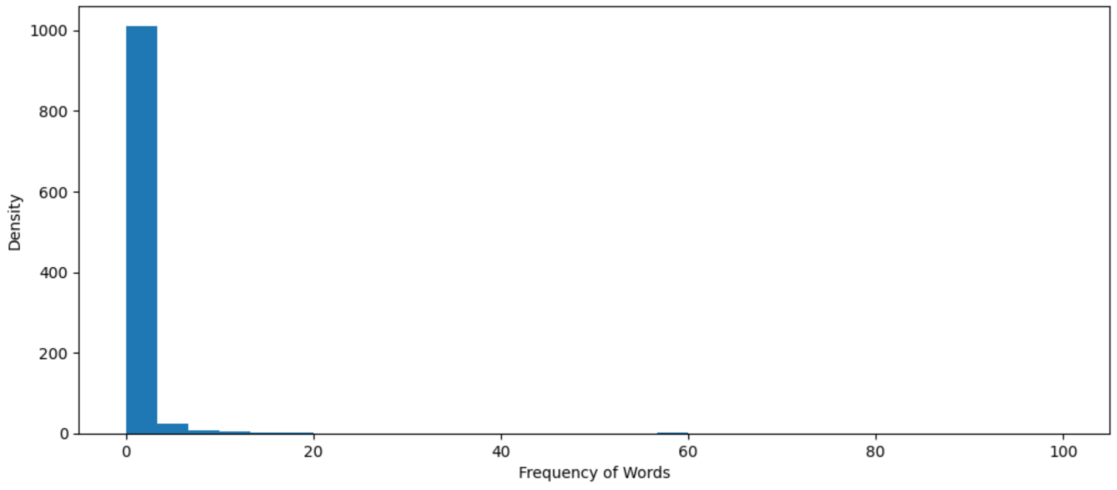
```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Glyph 48512 (\N{HANGUL SYLLABLE BU}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Glyph 50885 (\N{HANGUL SYLLABLE UNG}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



[84]:
```
sns.countplot(data=df, x='like_count')
plt.title('Sentiment Anaysis')
plt.show()
```



```
[72]: import matplotlib.pyplot as plt
      plt.figure(figsize=(12, 5))
      plt.hist(features_counts_df['counts'], bins=30, range=(0, 100))
      plt.xlabel('Frequency of Words')
      plt.ylabel('Density')
      plt.show()
```



```
[ ]:
```