Unraveling Personality Patterns on Netflix: Leveraging User Behavior and Surveys

Data Preprocessing

In [1]:

```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\supraja\anaconda3
\envs\myenv\lib\site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in c:\users\supraja\anaconda
3\envs\myenv\lib\site-packages (from openpyxl) (1.1.0)
```

In [2]:

```
import pandas as pd
data = pd.read_excel('Netflix Dataset Latest 2021.xlsx', index_col=[0])

```

In [3]:

```
data.head()
```

TO BUILD A GIRL	Comedy	Dramas,Comedies,Films Based on Books,British	English	Movie	7.0	
The Con-Heartist	Comedy, Romance	Romantic Comedies,Comedies,Romantic Films,Thai...	Thai	Movie	8.6	
Gleboka woda	Drama	TV Dramas,Polish TV Shows,Social Issue TV Dramas	Polish	Series	8.7	
Only a Mother	Drama	Social Issue Dramas,Dramas,Movies Based on Boo	Swedish	Movie	8.3	Lithuania,Poland,Fran

As we can clearly see, the dataset has null values and inconsistencies in it. So we need to drop the null columns and assign 0 to null in non-empty Colmns

In [4]:

```
data.dropna(axis=1, how='all',inplace=True)
data.fillna(0,inplace=True)
data.head()
```

Out[4]:

Title	Genre	Tags	Languages	Series or Movie	Hidden Gem Score	
Lets Fight Ghost	Crime, Drama, Fantasy, Horror, Romance	Comedy Programmes,Romantic TV Comedies,Horror ...	Swedish, Spanish	Series	4.3	
HOW TO BUILD A GIRL	Comedy	Dramas,Comedies,Films Based on Books,British	English	Movie	7.0	
The Con-Heartist	Comedy, Romance	Romantic Comedies,Comedies,Romantic Films,Thai...	Thai	Movie	8.6	
Gleboka woda	Drama	TV Dramas,Polish TV Shows,Social Issue TV Dramas	Polish	Series	8.7	
Only a Mother	Drama	Social Issue Dramas,Dramas,Movies Based on Boo...	Swedish	Movie	8.3	Lithuania,Pc

5 rows × 28 columns

Now the Dataset is free of null values and bull columns and the data seems more promosing for further Analysis

In [5]:

```
1 data.describe()
```

Out[5]:

	Hidden Gem Score	IMDb Score	Rotten Tomatoes Score	Metacritic Score	Awards Received	Awards Nominated For	
count	9425.000000	9425.000000	9425.000000	9425.000000	9425.000000	9425.000000	9
mean	5.534854	6.949613	37.373369	25.169125	5.398408	10.848064	1
std	2.452806	0.921831	37.281925	30.927880	15.322014	27.532930	5
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	3.400000	6.500000	0.000000	0.000000	0.000000	0.000000	0
50%	5.300000	7.000000	30.000000	0.000000	1.000000	2.000000	0
75%	8.100000	7.500000	75.000000	55.000000	4.000000	9.000000	8
max	9.800000	9.700000	100.000000	100.000000	300.000000	386.000000	6

Analysis

1.Logistic regression representation for IMDB votes V/s Awards Receving.

In [6]:

```
1  ## Importing Libraries
2
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  from sklearn.linear_model import LogisticRegression
7  from sklearn.model_selection import train_test_split
8  from sklearn.metrics import confusion_matrix, classification_report
9  from sklearn.metrics import mean_squared_error, r2_score
10
```

In [7]:

```
1 X = data['IMDb Votes'].values.reshape(-1, 1)
2 y = data['Awards Received'].values.reshape(-1, 1)
3
```

In [8]:

```
## Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [9]:

```
## Create and train the logistic regression model

model = LogisticRegression()
model.fit(X_train, y_train)

```

```
C:\Users\Supraja\anaconda3\envs\myenv\lib\site-packages\sklearn\utils
\validation.py:1111: DataConversionWarning: A column-vector y was pass
ed when a 1d array was expected. Please change the shape of y to (n_sa
mples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Supraja\anaconda3\envs\myenv\lib\site-packages\sklearn\linear
_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as show
n in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic
-regression (https://scikit-learn.org/stable/modules/linear_model.html
#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[9]:

▾ LogisticRegression

LogisticRegression()

In [10]:

```
## Make predictions using the trained model

y_pred = model.predict(X_test)

```

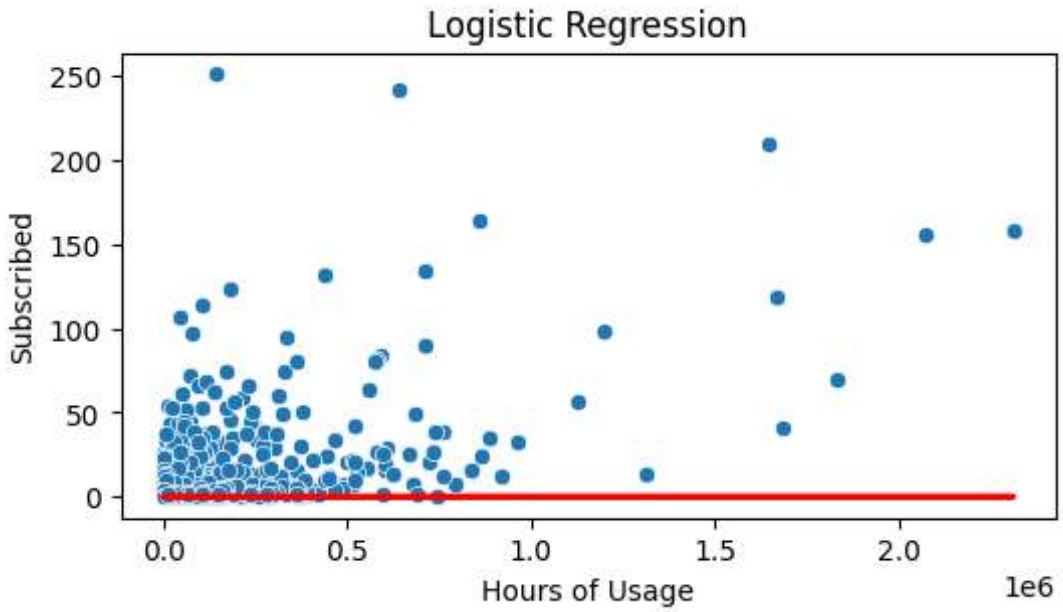
In [11]:

```
## Evaluate the model

conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print('Confusion Matrix:')
print(conf_matrix)
print('\nClassification Report:')
print(class_report)
```

```
       242.0       0.00      0.00      0.00         1
       251.0       0.00      0.00      0.00         1

    accuracy                           0.43      1885
   macro avg       0.01      0.01      0.01      1885
weighted avg       0.18      0.43      0.26      1885


C:\Users\Supraja\anaconda3\envs\myenv\lib\site-packages\sklearn
\metrics\_classification.py:1334: UndefinedMetricWarning: Preci
sion and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Supraja\anaconda3\envs\myenv\lib\site-packages\sklearn
\metrics\_classification.py:1334: UndefinedMetricWarning: Preci
sion and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
```

In [12]:

```
## Visualize the results

plt.figure(figsize=(6, 3))
sns.scatterplot(x=X_test.flatten(), y=y_test.flatten())
plt.plot(X_test.flatten(), y_pred.flatten(), color='red', linewidth=2)
plt.xlabel('Hours of Usage')
plt.ylabel('Subscribed')
plt.title('Logistic Regression')
plt.show()
```



In [88]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9425 entries, Lets Fight Ghost to DreamWorks Happy Holidays fro
m Madagascar
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Genre                  9400 non-null   object
 1   Tags                   9389 non-null   object
 2   Languages              9266 non-null   object
 3   Series or Movie        9425 non-null   object
 4   Hidden Gem Score       9415 non-null   float64
 5   Country Availability   9414 non-null   object
 6   Runtime                9424 non-null   object
 7   Director               7120 non-null   object
 8   Writer                 7615 non-null   object
 9   Actors                 9314 non-null   object
 10  View Rating            6827 non-null   object
 11  IMDb Score             9417 non-null   float64
 12  Rotten Tomatoes Score  5445 non-null   float64
 13  Metacritic Score       4082 non-null   float64
 14  Awards Received        5226 non-null   float64
 15  Awards Nominated For   6376 non-null   float64
 16  Boxoffice              3754 non-null   float64
 17  Release Date           9217 non-null   datetime64[ns]
 18  Netflix Release Date   9425 non-null   datetime64[ns]
 19  Production House       4393 non-null   object
 20  Netflix Link           9425 non-null   object
 21  IMDb Link              9101 non-null   object
 22  Summary                9420 non-null   object
 23  IMDb Votes             9415 non-null   float64
 24  Image                  9425 non-null   object
 25  Poster                 8487 non-null   object
 26  TMDb Trailer           9425 non-null   object
 27  Trailer Site           9424 non-null   object
dtypes: datetime64[ns](2), float64(8), object(18)
memory usage: 2.3+ MB
```

decision tree

2.What exactly represents the Netflix dataset's average geographic allocation for all viewing ratings?

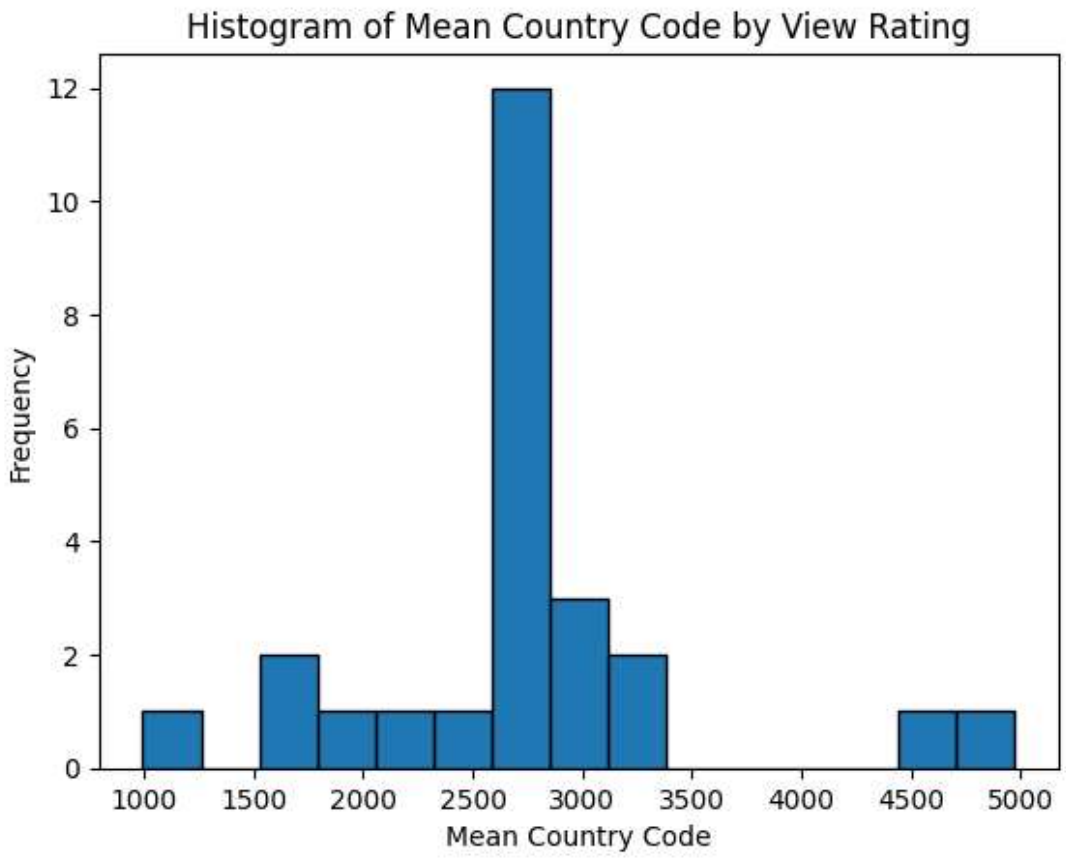
In [13]:

```
import pandas as pd

# Print the column names
print(data.columns)

```

```
Index(['Genre', 'Tags', 'Languages', 'Series or Movie', 'Hidden Gem Sc
ore',
       'Country Availability', 'Runtime', 'Director', 'Writer', 'Actor
s',
       'View Rating', 'IMDb Score', 'Rotten Tomatoes Score',
       'Metacritic Score', 'Awards Received', 'Awards Nominated For',
       'Boxoffice', 'Release Date', 'Netflix Release Date', 'Productio
n House',
       'Netflix Link', 'IMDb Link', 'Summary', 'IMDb Votes', 'Image',
'Poster',
       'TMDb Trailer', 'Trailer Site'],
      dtype='object')
```

In [22]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Create a label encoder
encoder = LabelEncoder()

# Encode the country names
data['Country Availability'] = data['Country Availability'].astype(str)
data['Country Code'] = encoder.fit_transform(data['Country Availability'])


# Group by 'View Rating' and calculate the mean of 'Country Code'
grouped_data = data.groupby('View Rating')['Country Code'].mean()

# Plot the histogram
plt.hist(grouped_data, bins=15, ec='black')

# Set the labels and title
plt.xlabel('Mean Country Code')
plt.ylabel('Frequency')
plt.title('Histogram of Mean Country Code by View Rating')

# Show the plot
plt.show()

```



Random Forest

```
1 #### 3.How could I use the Py WordCloud module to generate a word cloud visualization based on a DataFrame column?
```

In [23]:

```
1 !pip install --upgrade gensim
2
```

```
Requirement already satisfied: gensim in c:\users\supraja\anaconda3\en
vs\myenv\lib\site-packages (4.3.1)
Requirement already satisfied: numpy>=1.18.5 in c:\users\supraja\appda
ta\roaming\python\python310\site-packages (from gensim) (1.23.5)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\supraja\a
naconda3\envs\myenv\lib\site-packages (from gensim) (6.3.0)
Requirement already satisfied: scipy>=1.7.0 in c:\users\supraja\anacon
da3\envs\myenv\lib\site-packages (from gensim) (1.9.1)
```

In [24]:

```
1 !pip list
```

Package	Version
absl-py	1.3.0
alabaster	0.7.13
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.0.8
astunparse	1.6.3
async-generator	1.10
attrs	22.1.0
Automat	22.10.0
Babel	2.11.0
backcall	0.2.0
beautifulsoup4	4.11.1
bibtexparser	1.4.0
bleach	5.0.1
blis	0.7.9
branca	0.5.0
cachetools	5.2.0
catalogue	2.0.8
certifi	2022.9.24
cffi	1.15.1
charset-normalizer	2.1.1
click	8.1.3
colorama	0.4.6
confection	0.0.4
constantly	15.1.0
contourpy	1.0.5
cryptography	39.0.2
cssselect	1.2.0
cycler	0.11.0
cymem	2.0.7
datascience	0.17.5
debugpy	1.6.3
decorator	5.1.1
defusedxml	0.7.1
Deprecated	1.2.13
docutils	0.18.1
entrypoints	0.4
et-xmlfile	1.1.0
exceptiongroup	1.1.0
executing	1.1.0
fake-useragent	1.1.1
fastjsonschema	2.16.2
filelock	3.9.0
flatbuffers	22.11.23
folium	0.12.1.post1
fonttools	4.37.4
free-proxy	1.1.0
funcy	2.0
gast	0.4.0
gensim	4.3.1
google-auth	2.15.0
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
grpcio	1.51.1
h11	0.14.0
h5py	3.7.0
httpcore	0.16.3

httpx	0.23.3
huggingface-hub	0.13.0
hyperlink	21.0.0
idna	3.4
imagesize	1.4.1
incremental	22.10.0
ipykernel	6.16.0
ipython	8.5.0
ipython-genutils	0.2.0
ipywidgets	8.0.2
itemadapter	0.7.0
itemloaders	1.0.6
jedi	0.18.1
Jinja2	3.1.2
jmespath	1.0.1
joblib	1.2.0
jsonschema	4.16.0
jupyter	1.0.0
jupyter_client	7.3.5
jupyter-console	6.4.4
jupyter-core	4.11.1
jupyterlab-pygments	0.2.2
jupyterlab-widgets	3.0.3
keras	2.11.0
kiwisolver	1.4.4
langcodes	3.3.0
libclang	14.0.6
lxml	4.9.2
Markdown	3.4.1
MarkupSafe	2.1.1
matplotlib	3.6.0
matplotlib-inline	0.1.6
mistune	2.0.4
murmurhash	1.0.9
nbclient	0.6.8
nbconvert	7.1.0
nbformat	5.6.1
nest-asyncio	1.5.6
nltk	3.8.1
notebook	6.4.12
numexpr	2.8.4
numpy	1.23.5
oauthlib	3.2.2
openpyxl	3.1.2
opt-einsum	3.3.0
outcome	1.2.0
packaging	21.3
pandas	1.5.0
pandocfilters	1.5.0
parsel	1.7.0
parso	0.8.3
pathy	0.10.1
pickleshare	0.7.5
Pillow	9.2.0
pip	22.2.2
plotly	5.10.0
preshed	3.0.8
prometheus-client	0.14.1
prompt-toolkit	3.0.31
Protego	0.2.1
protobuf	3.19.6

```
psutil                        5.9.2
pure-eval                     0.2.2
pyasn1                        0.4.8
pyasn1-modules                0.2.8
pycparser                     2.21
pydantic                      1.10.7
PyDispatcher                  2.0.7
Pygments                      2.13.0
pyLDAvis                      3.4.0
pyOpenSSL                     23.0.0
pyparsing                     3.0.9
pyrsistent                    0.18.1
PySocks                       1.7.1
python-dateutil               2.8.2
python-dotenv                 0.21.1
pytz                          2022.4
pywin32                       304
pywinpty                      2.0.8
PyYAML                        6.0
pyzmq                         24.0.1
qtconsole                     5.3.2
QtPy                          2.2.1
queuelib                      1.6.2
regex                         2022.10.31
requests                      2.28.1
requests-file                 1.5.1
requests-oauthlib             1.3.1
rfc3986                       1.5.0
rsa                           4.9
scholarly                     1.7.11
scikit-learn                  1.1.3
scipy                         1.9.1
Scrapy                        2.8.0
seaborn                       0.12.1
selenium                      4.8.0
Send2Trash                    1.8.0
service-identity              21.1.0
setuptools                    65.6.3
six                           1.16.0
smart-open                    6.3.0
sniffio                       1.3.0
snowballstemmer               2.2.0
sortedcontainers              2.4.0
soupsieve                     2.3.2.post1
spacy                         3.5.1
spacy-legacy                  3.0.12
spacy-loggers                 1.0.4
Sphinx                        6.1.3
sphinx-rtd-theme              1.2.0
sphinxcontrib-applehelp       1.0.4
sphinxcontrib-devhelp         1.0.2
sphinxcontrib-htmlhelp        2.0.1
sphinxcontrib-jquery          2.0.0
sphinxcontrib-jsmath          1.0.1
sphinxcontrib-qthelp          1.0.3
sphinxcontrib-serializinghtml 1.1.5
srsly                         2.4.6
stack-data                    0.5.1
tenacity                      8.1.0
tensorboard                   2.11.0
tensorboard-data-server       0.6.1
```

In [85]:

```
print(data['Runtime'].head())
```

```
Title
Lets Fight Ghost       < 30 minutes
HOW TO BUILD A GIRL        1-2 hour
The Cone-Heart              > 2 hrs
Gleboka woda       < 30 minutes
Only a Mother              1-2 hour
Name: Runtime, dtype: object
```

```
tensorboard-plugin-wit        1.8.1
tensorflow                    2.11.0
tensorflow-estimator          2.11.0
tensorflow-intel              2.11.0
tensorflow-io-gcs-filesystem  0.28.0
termcolor                     2.1.1
terminado                     0.16.0
textblob                      0.17.1
thinc                         8.1.9
threadpoolctl                 3.1.0
tinycss2                      1.1.1
tldextract                    3.4.0
tokenizers                    0.13.2
torch                         1.13.1
torchaudio                    0.13.1+cu116
torchvision                   0.14.1+cu116
tornado                       6.2
tqdm                          4.65.0
traitlets                     5.1.0
transformers                  4.26.1
trio                          0.22.0
trio-websocket                0.9.2
Twisted                       22.10.0
twisted-iocpsupport           1.0.2
typer                         0.7.0
typing-extensions             4.4.0
urllib3                       1.26.13
wasabi                        1.1.1
wcwidth                       0.2.5
webencodings                  0.5.1
websocket-client              0.58.0
Werkzeug                      2.2.2
wheel                         0.38.4
widgetsnbextension            4.0.3
win-inet-pton                 1.1.0
wincertstore                  0.2
wordcloud                     1.9.2
wrapt                         1.14.1
wsproto                       1.2.0
xgboost                       1.7.5
zope.interface                5.5.2
```

In [26]:

```
data1 = data['View Rating'].values.tolist()
```

In [27]:

```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\supraja\anaconda3
\envs\myenv\lib\site-packages (1.9.2)
Requirement already satisfied: matplotlib in c:\users\supraja\anaconda
3\envs\myenv\lib\site-packages (from wordcloud) (3.6.0)
Requirement already satisfied: pillow in c:\users\supraja\anaconda3\en
vs\myenv\lib\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\supraja\appdat
a\roaming\python\python310\site-packages (from wordcloud) (1.23.5)
Requirement already satisfied: packaging>=20.0 in c:\users\supraja\app
data\roaming\python\python310\site-packages (from matplotlib->wordclou
d) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\supraja\ap
pdata\roaming\python\python310\site-packages (from matplotlib->wordclo
ud) (3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\supraja\an
aconda3\envs\myenv\lib\site-packages (from matplotlib->wordcloud) (1.
0.6)
Requirement already satisfied: cycler>=0.10 in c:\users\supraja\anacon
da3\envs\myenv\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\supraja\a
naconda3\envs\myenv\lib\site-packages (from matplotlib->wordcloud) (4.
38.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\supraj
a\anaconda3\envs\myenv\lib\site-packages (from matplotlib->wordcloud)
(2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\supraja\a
naconda3\envs\myenv\lib\site-packages (from matplotlib->wordcloud) (1.
4.4)
Requirement already satisfied: six>=1.5 in c:\users\supraja\appdata\ro
aming\python\python310\site-packages (from python-dateutil>=2.7->matpl
otlib->wordcloud) (1.16.0)
```

In [28]:

```
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from gensim.utils import simple_preprocess

stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use', 'not', 'would', 'say',

lemmatizer = WordNetLemmatizer()

def sent_to_words(sentences):
    for sent in sentences:
        sent = re.sub('\S*@\S*\s?', '', sent)  # remove emails
        sent = re.sub('\s+', ' ', sent)  # remove newline chars
        sent = re.sub("\'", "", sent)  # remove single quotes
        sent = simple_preprocess(str(sent), deacc=True)
        sent = [lemmatizer.lemmatize(word) for word in sent]  # lemmatize words
        yield(sent)

# Assuming 'data' is your DataFrame with a 'Languages' column
data = pd.DataFrame({'Languages': ['English', 'Spanish', 'French', 'German', 'It

# Assuming 'data' is your DataFrame with a 'Languages' column
data1 = data['Languages'].values.tolist()
data_words = list(sent_to_words(data1))

# Combine all words into a single string
all_words = ' '.join([' '.join(words) for words in data_words])

# Create WordCloud object
wordcloud = WordCloud(width=800, height=400, stopwords=stop_words, background_cc

# Display the word cloud
plt.figure(figsize=(8, 3))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()

```

Gaussian Naive Bayes

4.How could We estimate the "View Rating" according to "Tags" within the Netflix collection using a Naive Bayes classifier and CountVectorizer?

In [29]:

```
#Importing Liberaries

import pandas as pd
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
```

In [30]:

```
# Load the dataset into a pandas DataFrame
df = pd.read_excel('Netflix Dataset Latest 2021.xlsx')
```

In [31]:

```
# Training data
X_train = np.array(['Tags'])
y_train = np.array(['View Rating'])
```

In [32]:

```
# Test data
X_test = np.array(['View Rating'])
```

In [33]:

```
# Create a CountVectorizer to convert the text documents to a matrix of token co
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)
```

In [34]:

```
# Train a Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train_counts, y_train)
```

Out[34]:

MultinomialNB

MultinomialNB()

In [35]:

```
# Predict the sentiment of the test data
y_pred = classifier.predict(X_test_counts)
```

In [36]:

```
# Print the predicted labels
for doc, label in zip(X_test, y_pred):
    print(f"{doc} -> {label}")
```

View Rating -> View Rating

In [ ]:

```

```

In [ ]:

```

```