Analyzing Animes in the Netflix Dataset

By: Suryaa Rajinikanth

Introduction:

Hey, person judging my SIF application. My name is Suryaa Rajinikanth, and today I'll be using this Jupyter Notebook to outline some of the findings I came across while examining the Netflix Dataset. Let me preface this by saying I'm a newly converted anime fan, so when I saw the dataset about programs on Netflix, I knew that anime in the dataset is something I'd definitely be interested in exploring. Let me give you a list of what I want to accomplish with this dataset.

1. Compare show counts of animes vs similar genres
2. Display and analyze statistics on animes vs shows in general
3. Analyze the addition of animes throughout the years
4. Describe the countries and languages of animes vs similar genres
5. Find the best animes and see what makes them so good

Cool, let's get to it then.

General Anime Stats:

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

> **In this particular case, I decided to use pandas to efficiently parse data and matplotlib to visualize it.**

In [3]:
```
file = 'Netflix Dataset Latest 2021.xlsx'
netflix = pd.read_excel(file)
netflix.shape
```

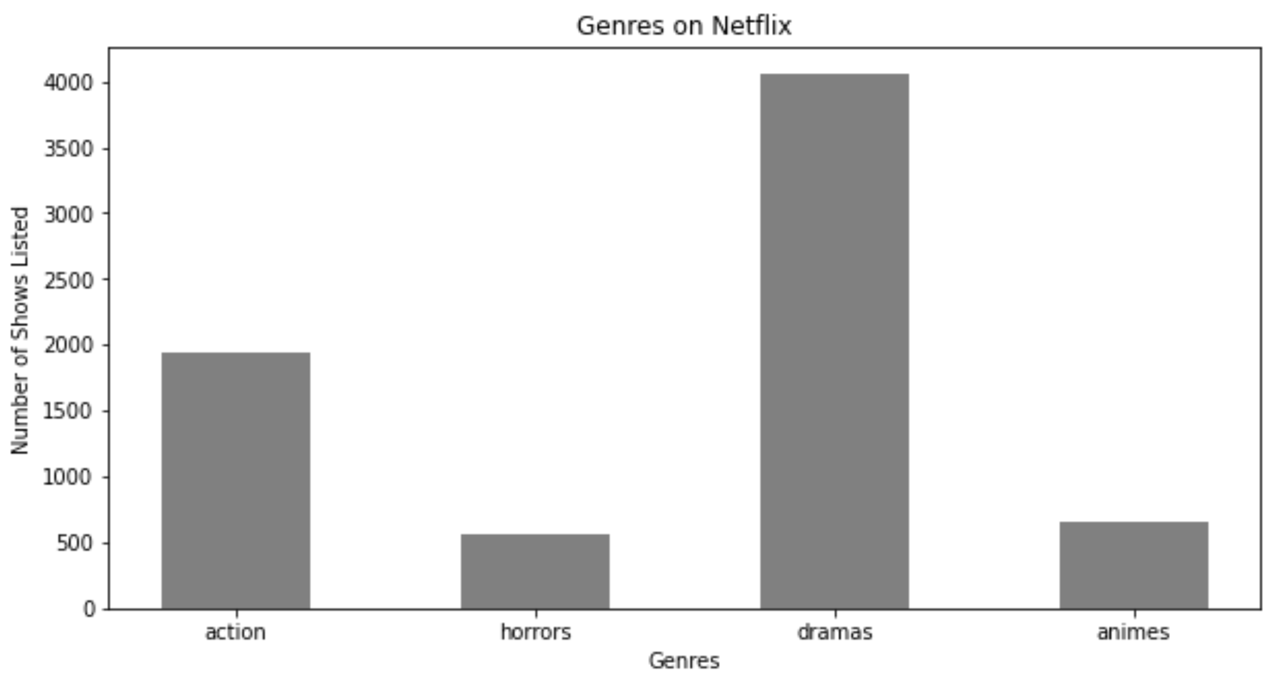
Out[3]: (9425, 29)

> **Wow, that's a lot of data. This means we have over 9k data points and 29 ways to gauge them.**

In [4]:
```
animes = netflix.loc[netflix['Tags'].str.contains("anime", case = False, na=
print(str(len(animes.index))+" animes are listed on Netflix.")
```

```
651 animes are listed on Netflix.
```

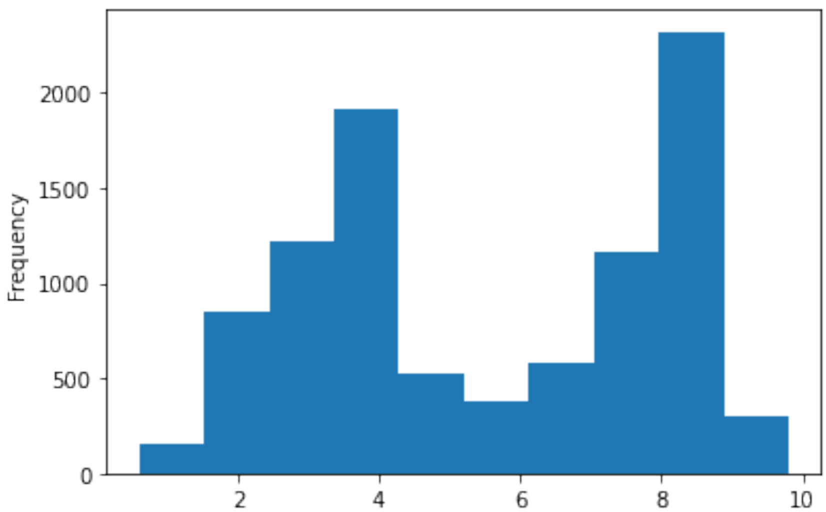
In [5]:
```
actions = netflix.loc[netflix['Tags'].str.contains("action", case = False, r
horrors = netflix.loc[netflix['Tags'].str.contains("horror", case = False, r
dramas = netflix.loc[netflix['Tags'].str.contains("drama", case = False, na=

data = {'action':len(actions.index), 'horrors':len(horrors.index), 'dramas':
courses = list(data.keys())
values = list(data.values())

figure = plt.figure(figsize = (10, 5))
plt.bar(courses, values, color ='grey', width = 0.5)
plt.xlabel("Genres")
plt.ylabel("Number of Shows Listed")
plt.title("Genres on Netflix")
plt.show()
```



In [6]:
```
netflix['Hidden Gem Score'].plot(kind="hist")
netflix.describe()
```

Out[6]:

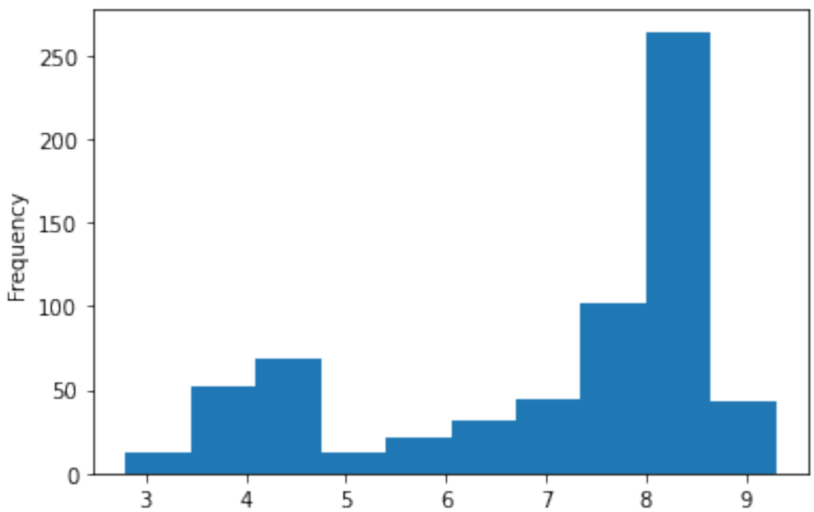
	Hidden Gem Score	IMDb Score	Rotten Tomatoes Score	Metacritic Score	Awards Received	Awards Nominated For	
count	9415.000000	9417.000000	5445.000000	4082.000000	5226.000000	6376.000000	3.
mean	5.540733	6.955517	64.691276	58.113425	9.735936	16.035602	4.
std	2.447462	0.899681	25.269466	17.143187	19.524116	32.209094	7.
min	0.600000	1.600000	0.000000	6.000000	1.000000	1.000000	7
25%	3.400000	6.500000	49.000000	46.000000	1.250000	2.000000	1.
50%	5.300000	7.000000	70.000000	59.000000	4.000000	6.000000	2.
75%	8.100000	7.500000	85.000000	71.000000	9.000000	15.000000	6
max	9.800000	9.700000	100.000000	100.000000	300.000000	386.000000	6.



In [7]:
```
animes['Hidden Gem Score'].plot(kind="hist")
animes.describe()
```

Out[7]:

	Hidden Gem Score	IMDb Score	Rotten Tomatoes Score	Metacritic Score	Awards Received	Awards Nominated For	Boxoff
count	651.000000	651.000000	127.000000	57.000000	157.000000	249.000000	8.500000e+
mean	7.039939	7.487404	80.496063	71.719298	3.783439	4.855422	1.689806e+
std	1.727293	0.609737	12.537483	11.766712	6.318794	8.439728	7.732988e+
min	2.800000	2.700000	43.000000	43.000000	1.000000	1.000000	6.461000e+
25%	6.000000	7.000000	73.000000	65.000000	1.000000	1.000000	1.721470e+
50%	7.800000	7.500000	82.000000	73.000000	2.000000	2.000000	4.981560e+
75%	8.300000	7.900000	89.500000	80.000000	3.000000	5.000000	2.250213e+
max	9.300000	9.100000	100.000000	96.000000	58.000000	69.000000	4.745447e+



Anime Specific Stats:

In [8]:

```
#animes['Release Date'] = pd.to_datetime(animes['Release Date'], format = '%
#animes.sort_values(by='Netflix Release Date')['Netflix Release Date'].dt.ye
#for row in df.rows:
```

Trying To Predict Score Using Description

> Ok, this has nothing to do with Anime. I just wanted to run some tests using the BERT model and Tensorflow to check if prediction of GEM score using a description works. I did something similar when I scraped Twitter for stock-related news for sentiment analysis, but this is the full, undistilled BERT.

In [9]:
```
! pip install bert-tensorflow
! pip install tensorflow-hub
! pip install tensorflow-datasets
```

```
Collecting bert-tensorflow
  Downloading bert_tensorflow-1.0.4-py2.py3-none-any.whl (64 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.4/64.4 kB 22.6 MB/s eta 0:
00:00
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from
bert-tensorflow) (1.14.0)
Installing collected packages: bert-tensorflow
Successfully installed bert-tensorflow-1.0.4
WARNING: Running pip as the 'root' user can result in broken permissions an
d conflicting behaviour with the system package manager. It is recommended
to use a virtual environment instead: https://pip.pypa.io/warnings/venv
Collecting tensorflow-hub
  Downloading tensorflow_hub-0.12.0-py2.py3-none-any.whl (108 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 108.8/108.8 kB 24.8 MB/s eta 0:
00:00
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.9/di
st-packages (from tensorflow-hub) (1.23.1)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.9/
dist-packages (from tensorflow-hub) (3.19.4)
Installing collected packages: tensorflow-hub
Successfully installed tensorflow-hub-0.12.0
WARNING: Running pip as the 'root' user can result in broken permissions an
d conflicting behaviour with the system package manager. It is recommended
to use a virtual environment instead: https://pip.pypa.io/warnings/venv
Collecting tensorflow-datasets
  Downloading tensorflow_datasets-4.6.0-py3-none-any.whl (4.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 90.6 MB/s eta 0:00
:00:00:01
Requirement already satisfied: absl-py in /usr/local/lib/python3.9/dist-pac
kages (from tensorflow-datasets) (1.1.0)
Collecting toml
  Downloading toml-0.10.2-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packa
ges (from tensorflow-datasets) (1.23.1)
Requirement already satisfied: protobuf>=3.12.2 in /usr/local/lib/python3.9
/dist-packages (from tensorflow-datasets) (3.19.4)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from
```

```
tensorflow-datasets) (1.14.0)
Requirement already satisfied: termcolor in /usr/local/lib/python3.9/dist-p
ackages (from tensorflow-datasets) (1.1.0)
Collecting tensorflow-metadata
  Downloading tensorflow_metadata-1.10.0-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.8/50.8 kB 19.0 MB/s eta 0:
00:00
Requirement already satisfied: etils[epath] in /usr/local/lib/python3.9/dis
t-packages (from tensorflow-datasets) (0.6.0)
Collecting promise
  Downloading promise-2.3.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: dill in /usr/local/lib/python3.9/dist-packag
es (from tensorflow-datasets) (0.3.5.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packag
es (from tensorflow-datasets) (4.64.0)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.9
/dist-packages (from tensorflow-datasets) (2.28.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3/dist-
packages (from requests>=2.19.0->tensorflow-datasets) (2019.11.28)
Requirement already satisfied: idna<4,>=2.5 in /usr/lib/python3/dist-packag
es (from requests>=2.19.0->tensorflow-datasets) (2.8)
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/p
ython3.9/dist-packages (from requests>=2.19.0->tensorflow-datasets) (2.1.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/pyth
on3.9/dist-packages (from requests>=2.19.0->tensorflow-datasets) (1.26.10)
Requirement already satisfied: zipp in /usr/local/lib/python3.9/dist-packag
es (from etils[epath]->tensorflow-datasets) (3.8.1)
Requirement already satisfied: importlib_resources in /usr/local/lib/python
3.9/dist-packages (from etils[epath]->tensorflow-datasets) (5.8.0)
Collecting googleapis-common-protos<2,>=1.52.0
  Downloading googleapis_common_protos-1.56.4-py2.py3-none-any.whl (211 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.7/211.7 kB 59.2 MB/s eta 0:
00:00
Requirement already satisfied: typing_extensions in /usr/local/lib/python3.
9/dist-packages (from etils[epath]->tensorflow-datasets) (4.3.0)
Building wheels for collected packages: promise
  Building wheel for promise (setup.py) ... done
  Created wheel for promise: filename=promise-2.3-py3-none-any.whl size=214
86 sha256=623581a0368f80433f3717fc18b6b3d17e77950fdbb6c15ce22088324a1405fb
  Stored in directory: /root/.cache/pip/wheels/e1/e8/83/ddea66100678d139b14
bc87692ece57c6a2a937956d2532608
Successfully built promise
Installing collected packages: toml, promise, googleapis-common-protos, ten
sorflow-metadata, tensorflow-datasets
Successfully installed googleapis-common-protos-1.56.4 promise-2.3 tensorfl
ow-datasets-4.6.0 tensorflow-metadata-1.10.0 toml-0.10.2
WARNING: Running pip as the 'root' user can result in broken permissions an
d conflicting behaviour with the system package manager. It is recommended
to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [10]:
```
import re
import datetime
import numpy as np
from collections import Counter
import fractions
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import multilabel_confusion_matrix
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_datasets as tfds
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import classification_report

netflix1 = netflix[["Genre", "Summary"]]
genres = list()
for index, row in netflix1.iterrows():
  inner_genres = (str(row['Genre'])).split(", ")
  for inner_genre in inner_genres:
    genres.append(inner_genre)
true_genres = np.unique(genres)

len(true_genres)
```

Out[10]: 29

In [11]:
```
genrerow = [str(row.Genre).split(", ") for index, row in netflix1.iterrows()
mylist = list()
for i, genre in enumerate(true_genres):
  mylist.append([genre in mgenres for mgenres in genrerow])
print(len(mylist))
array = np.array(mylist).transpose()
print(len(array))
netflix2 = pd.DataFrame(data=array.astype(np.int64), columns=true_genres)
netflix2["Summary"] = netflix1.Summary
(netflix2.head(5))
```

```
29
9425
```

Out[11]:

	Action	Adult	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama
0	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1

5 rows × 30 columns

In [12]:
```
row_count = len(netflix2)
test_row_count = round(0.05 * row_count)

test_row = [i for i in np.arange(0,row_count,test_row_count)]
train_row = list(set([i for i in range(row_count)]) - set(test_row))

tdf = netflix2.iloc[test_row,:]
tensor_te = tf.data.Dataset.from_tensor_slices((tdf.Summary.values, tdf.drop
trdf = netflix2.iloc[test_row,:]
tensor_tr = tf.data.Dataset.from_tensor_slices((trdf.Summary.values, trdf.dr
tensor_train = tensor_tr.batch(32)
tensor_test = tensor_te.batch(32)

embedding_layer = hub.KerasLayer("https://tfhub.dev/google/tf2-preview/nnlm-
embedding_layer(netflix2.iloc[test_row,:]["Summary"][:1].to_numpy())

BERT = tf.keras.Sequential()
BERT.add(embedding_layer)
BERT.add(tf.keras.layers.Dense(64, activation='relu'))
BERT.add(tf.keras.layers.Dense(len(true_genres), activation='sigmoid'))

BERT.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 128)               124642688

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 29)                1885

=================================================================
Total params: 124,652,829
Trainable params: 10,141
Non-trainable params: 124,642,688
_________________________________________________________________
```

In [18]:
```
def macro_double_soft_f1(y, y_hat):
    """Compute the macro soft F1-score as a cost (average 1 - soft-F1 across
    Use probability values instead of binary predictions.
    This version uses the computation of soft-F1 for both positive and negat

    Args:
        y (int32 Tensor): targets array of shape (BATCH_SIZE, N_LABELS)
        y_hat (float32 Tensor): probability matrix from forward propagation

    Returns:
        cost (scalar Tensor): value of the cost function for the batch
    """
    y = tf.cast(y, tf.float32)
    y_hat = tf.cast(y_hat, tf.float32)
    tp = tf.reduce_sum(y_hat * y, axis=0)
    fp = tf.reduce_sum(y_hat * (1 - y), axis=0)
    fn = tf.reduce_sum((1 - y_hat) * y, axis=0)
    tn = tf.reduce_sum((1 - y_hat) * (1 - y), axis=0)
    soft_f1_class1 = 2*tp / (2*tp + fn + fp + 1e-16)
    soft_f1_class0 = 2*tn / (2*tn + fn + fp + 1e-16)
    cost_class1 = 1 - soft_f1_class1 # reduce 1 - soft-f1_class1 in order to
    cost_class0 = 1 - soft_f1_class0 # reduce 1 - soft-f1_class0 in order to
    cost = 0.5 * (cost_class1 + cost_class0) # take into account both class
    macro_cost = tf.reduce_mean(cost) # average on all labels
    return macro_cost

# Setup Tensorboard
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histo
!rm -rf ./logs/ # Clear any logs from previous runs

# Fit Network
BERT.compile(optimizer="adam", loss=macro_double_soft_f1, \
             metrics=[tf.keras.metrics.Accuracy(), tf.keras.metrics.Precisi
                      tf.keras.metrics.Recall()]) # "categorical_crossentro
history = BERT.fit(tensor_train, validation_data=tensor_test, epochs=25, ver
```

```
Epoch 1/25
1/1 [==============================] - 10s 10s/step - loss: 0.6000 - accura
cy: 0.0000e+00 - precision_1: 0.1951 - recall_1: 0.4000 - val_loss: 0.5986
- val_accuracy: 0.0000e+00 - val_precision_1: 0.2119 - val_recall_1: 0.4167
Epoch 2/25
1/1 [==============================] - 9s 9s/step - loss: 0.5986 - accuracy
: 0.0000e+00 - precision_1: 0.2119 - recall_1: 0.4167 - val_loss: 0.5972 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2273 - val_recall_1: 0.4167
Epoch 3/25
1/1 [==============================] - 9s 9s/step - loss: 0.5972 - accuracy
: 0.0000e+00 - precision_1: 0.2273 - recall_1: 0.4167 - val_loss: 0.5957 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2381 - val_recall_1: 0.4167
Epoch 4/25
```

```
1/1 [==============================] - 9s 9s/step - loss: 0.5957 - accuracy
: 0.0000e+00 - precision_1: 0.2381 - recall_1: 0.4167 - val_loss: 0.5942 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2708 - val_recall_1: 0.4333
Epoch 5/25
1/1 [==============================] - 9s 9s/step - loss: 0.5942 - accuracy
: 0.0000e+00 - precision_1: 0.2708 - recall_1: 0.4333 - val_loss: 0.5926 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2857 - val_recall_1: 0.4333
Epoch 6/25
1/1 [==============================] - 9s 9s/step - loss: 0.5926 - accuracy
: 0.0000e+00 - precision_1: 0.2857 - recall_1: 0.4333 - val_loss: 0.5910 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2889 - val_recall_1: 0.4333
Epoch 7/25
1/1 [==============================] - 9s 9s/step - loss: 0.5910 - accuracy
: 0.0000e+00 - precision_1: 0.2889 - recall_1: 0.4333 - val_loss: 0.5894 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.2955 - val_recall_1: 0.4333
Epoch 8/25
1/1 [==============================] - 10s 10s/step - loss: 0.5894 - accura
cy: 0.0000e+00 - precision_1: 0.2955 - recall_1: 0.4333 - val_loss: 0.5877
- val_accuracy: 0.0000e+00 - val_precision_1: 0.3095 - val_recall_1: 0.4333
Epoch 9/25
1/1 [==============================] - 9s 9s/step - loss: 0.5877 - accuracy
: 0.0000e+00 - precision_1: 0.3095 - recall_1: 0.4333 - val_loss: 0.5860 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3253 - val_recall_1: 0.4500
Epoch 10/25
1/1 [==============================] - 9s 9s/step - loss: 0.5860 - accuracy
: 0.0000e+00 - precision_1: 0.3253 - recall_1: 0.4500 - val_loss: 0.5842 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3333 - val_recall_1: 0.4333
Epoch 11/25
1/1 [==============================] - 9s 9s/step - loss: 0.5842 - accuracy
: 0.0000e+00 - precision_1: 0.3333 - recall_1: 0.4333 - val_loss: 0.5824 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3288 - val_recall_1: 0.4000
Epoch 12/25
1/1 [==============================] - 9s 9s/step - loss: 0.5824 - accuracy
: 0.0000e+00 - precision_1: 0.3288 - recall_1: 0.4000 - val_loss: 0.5806 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3333 - val_recall_1: 0.4000
Epoch 13/25
1/1 [==============================] - 9s 9s/step - loss: 0.5806 - accuracy
: 0.0000e+00 - precision_1: 0.3333 - recall_1: 0.4000 - val_loss: 0.5787 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3380 - val_recall_1: 0.4000
Epoch 14/25
1/1 [==============================] - 9s 9s/step - loss: 0.5787 - accuracy
: 0.0000e+00 - precision_1: 0.3380 - recall_1: 0.4000 - val_loss: 0.5768 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3284 - val_recall_1: 0.3667
Epoch 15/25
1/1 [==============================] - 9s 9s/step - loss: 0.5768 - accuracy
: 0.0000e+00 - precision_1: 0.3284 - recall_1: 0.3667 - val_loss: 0.5748 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3333 - val_recall_1: 0.3667
Epoch 16/25
1/1 [==============================] - 9s 9s/step - loss: 0.5748 - accuracy
: 0.0000e+00 - precision_1: 0.3333 - recall_1: 0.3667 - val_loss: 0.5728 -
```

```
val_accuracy: 0.0000e+00 - val_precision_1: 0.3333 - val_recall_1: 0.3667
Epoch 17/25
1/1 [==============================] - 9s 9s/step - loss: 0.5728 - accuracy
: 0.0000e+00 - precision_1: 0.3333 - recall_1: 0.3667 - val_loss: 0.5707 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3281 - val_recall_1: 0.3500
Epoch 18/25
1/1 [==============================] - 9s 9s/step - loss: 0.5707 - accuracy
: 0.0000e+00 - precision_1: 0.3281 - recall_1: 0.3500 - val_loss: 0.5686 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3443 - val_recall_1: 0.3500
Epoch 19/25
1/1 [==============================] - 9s 9s/step - loss: 0.5686 - accuracy
: 0.0000e+00 - precision_1: 0.3443 - recall_1: 0.3500 - val_loss: 0.5664 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3509 - val_recall_1: 0.3333
Epoch 20/25
1/1 [==============================] - 9s 9s/step - loss: 0.5664 - accuracy
: 0.0000e+00 - precision_1: 0.3509 - recall_1: 0.3333 - val_loss: 0.5643 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3725 - val_recall_1: 0.3167
Epoch 21/25
1/1 [==============================] - 9s 9s/step - loss: 0.5643 - accuracy
: 0.0000e+00 - precision_1: 0.3725 - recall_1: 0.3167 - val_loss: 0.5620 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3725 - val_recall_1: 0.3167
Epoch 22/25
1/1 [==============================] - 9s 9s/step - loss: 0.5620 - accuracy
: 0.0000e+00 - precision_1: 0.3725 - recall_1: 0.3167 - val_loss: 0.5598 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3800 - val_recall_1: 0.3167
Epoch 23/25
1/1 [==============================] - 9s 9s/step - loss: 0.5598 - accuracy
: 0.0000e+00 - precision_1: 0.3800 - recall_1: 0.3167 - val_loss: 0.5575 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3696 - val_recall_1: 0.2833
Epoch 24/25
1/1 [==============================] - 9s 9s/step - loss: 0.5575 - accuracy
: 0.0000e+00 - precision_1: 0.3696 - recall_1: 0.2833 - val_loss: 0.5551 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.3953 - val_recall_1: 0.2833
Epoch 25/25
1/1 [==============================] - 9s 9s/step - loss: 0.5551 - accuracy
: 0.0000e+00 - precision_1: 0.3953 - recall_1: 0.2833 - val_loss: 0.5527 -
val_accuracy: 0.0000e+00 - val_precision_1: 0.4250 - val_recall_1: 0.2833
```

In [23]:

```
y_test = np.concatenate([ex[1].numpy() for ex in tensor_test])
pre_y = BERT.predict(tensor_test).round().astype(np.int64)
cm_matrices = multilabel_confusion_matrix(y_test, pre_y)
print(classification_report(y_test, pre_y, target_names=true_genres))
```

```
1/1 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

      Action       0.00      0.00      0.00         1
       Adult       0.00      0.00      0.00         0
   Adventure       0.00      0.00      0.00         5
   Animation       0.00      0.00      0.00         3
   Biography       0.00      0.00      0.00         0
      Comedy       1.00      0.38      0.55         8
       Crime       0.00      0.00      0.00         2
 Documentary       0.00      0.00      0.00         1
       Drama       0.89      0.67      0.76        12
      Family       1.00      0.43      0.60         7
     Fantasy       0.00      0.00      0.00         7
   Film-Noir       0.00      0.00      0.00         0
   Game-Show       0.00      0.00      0.00         0
     History       1.00      1.00      1.00         1
      Horror       0.00      0.00      0.00         1
       Music       0.00      0.00      0.00         0
     Musical       0.00      0.00      0.00         1
     Mystery       0.00      0.00      0.00         0
        News       0.00      0.00      0.00         0
  Reality-TV       0.00      0.00      0.00         0
     Romance       1.00      0.25      0.40         8
      Sci-Fi       0.00      0.00      0.00         0
       Short       0.00      0.00      0.00         1
       Sport       0.00      0.00      0.00         0
   Talk-Show       0.00      0.00      0.00         0
    Thriller       0.00      0.00      0.00         1
         War       0.00      0.00      0.00         1
     Western       0.00      0.00      0.00         0
         nan       0.00      0.00      0.00         0

   micro avg       0.42      0.28      0.34        60
   macro avg       0.17      0.09      0.11        60
weighted avg       0.58      0.28      0.37        60
 samples avg       0.37      0.29      0.28        60
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1
327: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1
327: UndefinedMetricWarning: Recall and F-score are ill-defined and being s
et 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))
```