On the track of the most important movie feature

February 4, 2022

1 Authors: Natalia Czerep, Anna Pacanowska

2 Objective

The aim of our project was to create a model that predicts the movie genres based on other features, such as keywords, title, budget etc.

3 Dataset

3.1 Description

We used a dataset from Kaggle (https://www.kaggle.com/rounakbanik/the-movies-dataset). It contains metadata for 45 000 movies. There are many different features in the dataset - we were interested in the 'genres', 'budget', 'production_companies', 'production_countries', 'revenue', 'title', 'cast', 'director' and 'keywords'. Each movie can have multiple genres.

3.2 Data

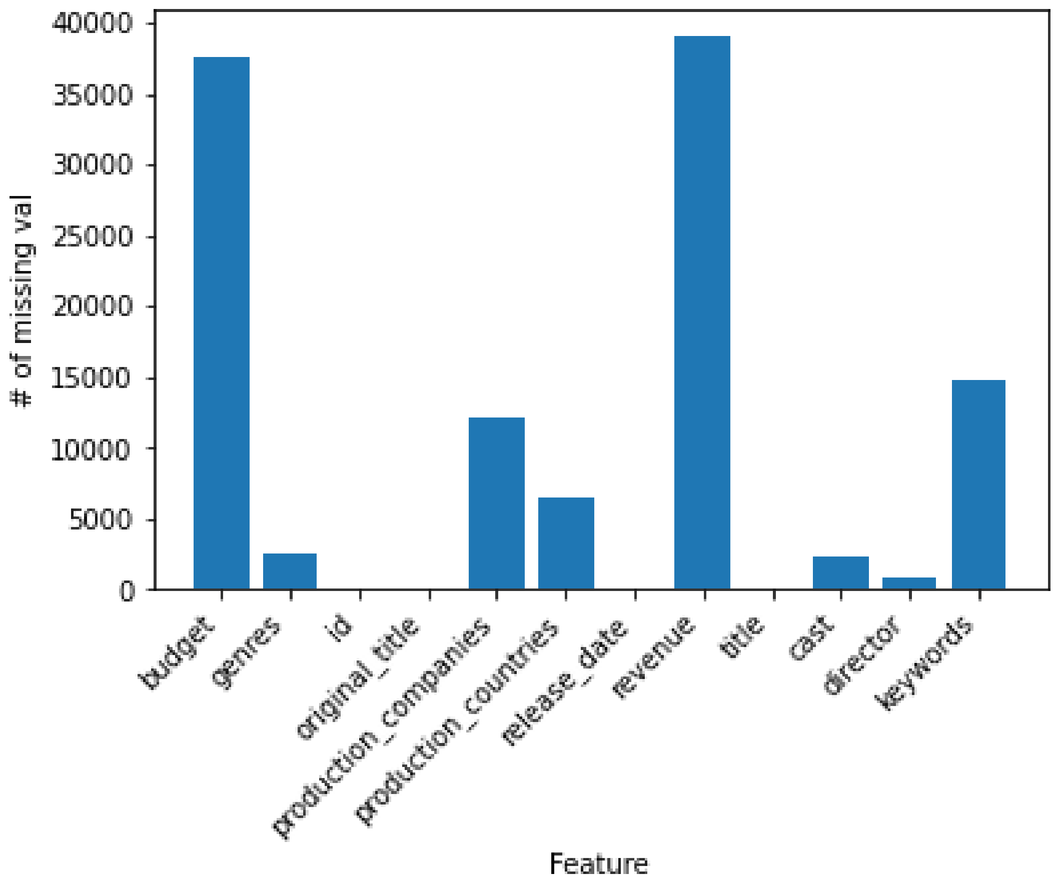
It turned out that many samples had missing values, e.g. there were 14818 samples with empty keywords list or 12199 with no production companies. The columns 'budget' and 'revenue' were set to 0 over 3700 times - this is most of the dataset. There were a few rows with data of incorrect type, such as strings in the column 'budget', which we dropped. The genres were not balanced - the most popular, 'Drama' occured over 20000 times, and the least popular - TV Movie - only around 1000 times.

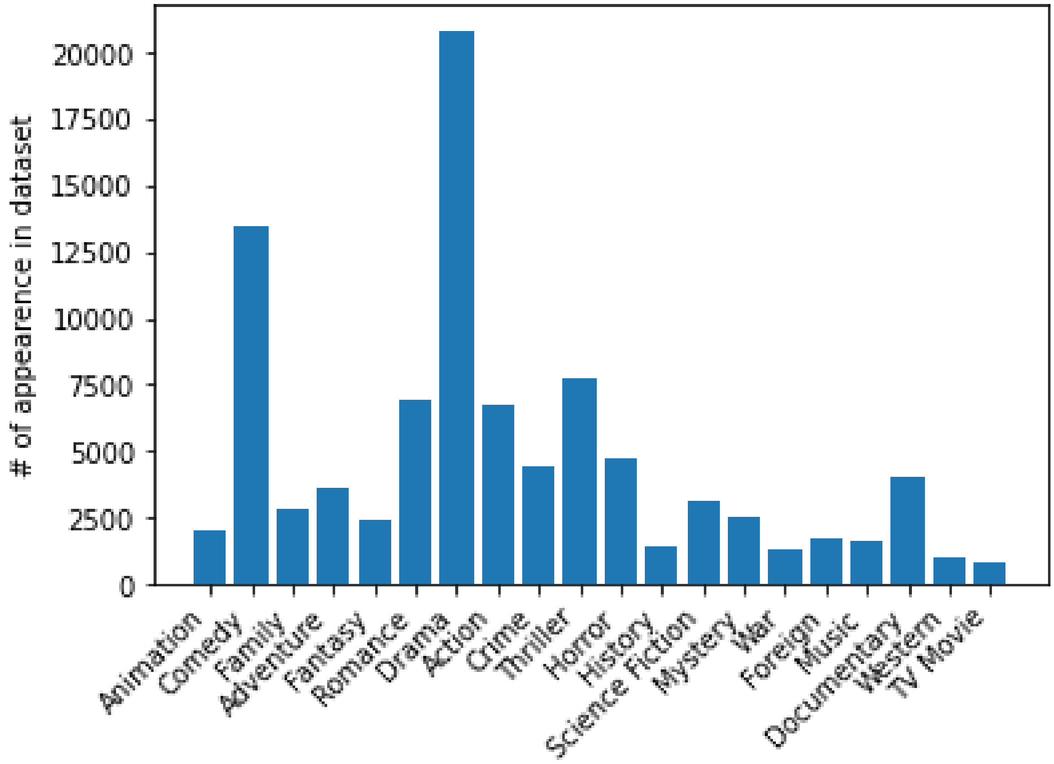
The figures showing detailed statistics and the most common combinations of genres are presented below.

[1]:
```
%run data_prep.ipynb
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\pandas\core\indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```


40000
35000
30000
25000
20000
15000
10000
5000
0
# of missing val
budget
genres
id
original_title
production_companies
production_countries
release_date
revenue
title
cast
director
keywords
Feature




```
Animation :  [(878, 'Family'), (564, 'Comedy')]
Comedy :  [(4295, 'Drama'), (3197, 'Romance')]
Family :  [(1205, 'Comedy'), (878, 'Animation')]
Adventure :  [(1775, 'Action'), (1071, 'Drama')]
Fantasy :  [(744, 'Comedy'), (712, 'Drama')]
Romance :  [(4605, 'Drama'), (3197, 'Comedy')]
Drama :  [(4605, 'Romance'), (4295, 'Comedy')]
Action :  [(2395, 'Thriller'), (2368, 'Drama')]
Crime :  [(2581, 'Drama'), (2055, 'Thriller')]
Thriller :  [(3491, 'Drama'), (2395, 'Action')]
Horror :  [(1951, 'Thriller'), (890, 'Science Fiction')]
History :  [(1098, 'Drama'), (346, 'War')]
Science Fiction :  [(1101, 'Action'), (890, 'Horror')]
Mystery :  [(1535, 'Thriller'), (1201, 'Drama')]
War :  [(1006, 'Drama'), (346, 'History')]
Foreign :  [(1009, 'Drama'), (408, 'Comedy')]
Music :  [(645, 'Drama'), (578, 'Comedy')]
Documentary :  [(348, 'Music'), (216, 'Drama')]
Western :  [(371, 'Action'), (277, 'Drama')]
TV Movie :  [(406, 'Drama'), (174, 'Comedy')]
```

3.3 Multi label and text features

Many of the columns were text, multi-label or both. We tried two distinct solutions for representing them.

First, we represented multi-label columns as binary matrixes - each column represented one label and was set to 1 if the sample contained this label. This is the standard representation in sklearn. The features could have many unique labels - e.g. there were 18186 different keywords. We used only the popular labels, e.g. keywords that occured at least 20 times, and discarded the rest.

However, this solution could not work for the text values that are unique for each row, such as the titles. Therefore we tried another solution - we used a pre-trained (Wikipedia 2014 + Gigaword) Word2Vec model. We took the average of the vectors for each word in the value.

3.4 Handling missing values

We used two different solutions for handling the missing data. First, we simply treated them as normal values. Second, we tried imputing them using IterativeImputer from sklearn.

4 Survey

In order to gauge the difficulty of the task we created a small survey. We selected 11 movies from the dataset and asked the participants to guess the genres based on the title and keywords. There were 15 responses. We calculated the averge precision to be 0.382 and and the average recall to be 0.421.

5 Models

We used three different combinations of the data:
(1) Only numeric columns: 'budget' and 'revenue'
(2) Columns 'keywords' and 'production_countries', limited to the most popular labels and processed by MultiLabelBinarizer + 'budget', 'revenue'
(3) Columns 'keywords' and 'titles' transformed by Word2Vec and avereged + 'production_companies_cols', 'production_countries_cols', 'cast' limited to the most popular and processed by MultiLabelBinarizer + 'budget', 'revenue'

We used three different models: logistic regression, random forest and XGBoost.

In order to score the performance of the model, we use classification_report from sklearn. It calculates precision, recall and the F1 score. ## Logistic regression The logistic regression did not perform well. For the data (2) it gave the following results:

	Precision	Recall	F1 Score
micro avg	0.111	0.017	0.030
macro avg	0.033	0.017	0.020
weighted avg	0.058	0.017	0.024
samples avg	0.006	0.015	0.008

5.1 Random forest

The random forest was much better. It performed best for the data (2):

	Precision	Recall	F1 Score
micro avg	0.601	0.260	0.363
macro avg	0.546	0.189	0.273
weighted avg	0.576	0.260	0.346
samples avg	0.360	0.263	0.284

5.2 XGBoost

The best results in this project were achieved using this model and the data (3):

	Precision	Recall	F1 Score
micro avg	0.670	0.351	0.461
macro avg	0.683	0.257	0.356
weighted avg	0.666	0.351	0.440
samples avg	0.503	0.384	0.406

5.3 Order of imputation

We tried different strategies for imputation_order available for IterativeImputer - ‘ascending’, ‘descending’, ‘roman’, ‘arabic’, ‘random’. The differences between the strategies were not large. The best strategy of ordering differed depending on the metric and model used.

All results for different combinations of the data and models can be found in the code section.

6 Code

Below is the full code for our project along the outputs.

data_prep

February 4, 2022

[28]:
```python
import pandas as pd
import ast
import matplotlib.pyplot as plt
from collections import defaultdict as dd
```

[4]:
```python
# Data from https://www.kaggle.com/rounakbanik/the-movies-dataset
movies_cols = ['genres', 'id', 'budget', 'original_title',␣
 ↪'production_companies', 'production_countries',
               'release_date', 'revenue', 'title']
movies_df = pd.read_csv("movies_metadata.csv.zip", usecols=movies_cols).dropna()
credits_df = pd.read_csv("credits.csv.zip")
keywords_df = pd.read_csv("keywords.csv.zip")
```

[5]:
```python
movies_df.head()
```

[5]:
```
     budget                                             genres     id  \
0  30000000  [{'id': 16, 'name': 'Animation'}, {'id': 35, '…    862
1  65000000  [{'id': 12, 'name': 'Adventure'}, {'id': 14, '…   8844
2         0  [{'id': 10749, 'name': 'Romance'}, {'id': 35, …  15602
3  16000000  [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam…  31357
4         0                     [{'id': 35, 'name': 'Comedy'}]  11862

                original_title  \
0                    Toy Story
1                      Jumanji
2             Grumpier Old Men
3            Waiting to Exhale
4  Father of the Bride Part II

                                production_companies  \
0     [{'name': 'Pixar Animation Studios', 'id': 3}]
1  [{'name': 'TriStar Pictures', 'id': 559}, {'na…
2  [{'name': 'Warner Bros.', 'id': 6194}, {'name'…
3  [{'name': 'Twentieth Century Fox Film Corporat…
4  [{'name': 'Sandollar Productions', 'id': 5842}…

                                production_countries release_date  \
```

```
0  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-10-30
1  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-15
2  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-22
3  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-22
4  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-02-10

       revenue                         title
0  373554033.0                     Toy Story
1  262797249.0                       Jumanji
2          0.0              Grumpier Old Men
3   81452156.0             Waiting to Exhale
4   76578911.0  Father of the Bride Part II
```

[6]:
```python
credits_df.head()
```

[6]:
```
                                                cast  \
0  [{'cast_id': 14, 'character': 'Woody (voice)',…
1  [{'cast_id': 1, 'character': 'Alan Parrish', '…
2  [{'cast_id': 2, 'character': 'Max Goldman', 'c…
3  [{'cast_id': 1, 'character': "Savannah 'Vannah…
4  [{'cast_id': 1, 'character': 'George Banks', '…

                                                crew     id
0  [{'credit_id': '52fe4284c3a36847f8024f49', 'de…    862
1  [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de…   8844
2  [{'credit_id': '52fe466a9251416c75077a89', 'de…  15602
3  [{'credit_id': '52fe44779251416c91011acb', 'de…  31357
4  [{'credit_id': '52fe44959251416c75039ed7', 'de…  11862
```

[7]:
```python
keywords_df.head()
```

[7]:
```
      id                                           keywords
0    862  [{'id': 931, 'name': 'jealousy'}, {'id': 4290,…
1   8844  [{'id': 10090, 'name': 'board game'}, {'id': 1…
2  15602  [{'id': 1495, 'name': 'fishing'}, {'id': 12392…
3  31357  [{'id': 818, 'name': 'based on novel'}, {'id':…
4  11862  [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n…
```

[8]:
```python
def resolve_jsons(df, column_name, ):
    i = 0
    for col in df[column_name]:
        new_col = []
        try:
            col = ast.literal_eval(col)
            for c in col:
                new_col.append(c["name"])
            df[column_name].iloc[i] = new_col
```

```
        except (ValueError,TypeError):
            pass
        i+=1
```

[9]:
```
resolve_jsons(movies_df, 'genres' )
resolve_jsons(movies_df, 'production_companies')
resolve_jsons(movies_df, 'production_countries')
resolve_jsons(keywords_df, 'keywords')
resolve_jsons(credits_df,'cast')
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\pandas\core\indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

[10]:
```
#isolated director from crew property and repleced the column 'crew' for
 ↪'director'
i = 0
for col in credits_df["crew"]:
    new_col = []
    try:
            col = ast.literal_eval(col)
            for c in col:
                if c["job"] == "Director":
                    new_col.append(c["name"])
            credits_df["crew"].iloc[i] = new_col
    except (ValueError,TypeError):
        pass
    i+=1
credits_df = credits_df.rename(columns={"crew":"director"})
```

[11]:
```
credits_df.head()
```

[11]:

	cast	director	id
0	[Tom Hanks, Tim Allen, Don Rickles, Jim Varney…	[John Lasseter]	862
1	[Robin Williams, Jonathan Hyde, Kirsten Dunst,…	[Joe Johnston]	8844
2	[Walter Matthau, Jack Lemmon, Ann-Margret, Sop…	[Howard Deutch]	15602
3	[Whitney Houston, Angela Bassett, Loretta Devi…	[Forest Whitaker]	31357
4	[Steve Martin, Diane Keaton, Martin Short, Kim…	[Charles Shyer]	11862

[12]:
```
#merging credits,movies,keywords df into one table
credits_df = credits_df.merge(keywords_df,how='right',on='id')
```

[13]:
```
credits_df.head()
```

[13]:
```
                                                cast           director  \
0  [Tom Hanks, Tim Allen, Don Rickles, Jim Varney…    [John Lasseter]
1  [Robin Williams, Jonathan Hyde, Kirsten Dunst,…     [Joe Johnston]
2  [Walter Matthau, Jack Lemmon, Ann-Margret, Sop…    [Howard Deutch]
3  [Whitney Houston, Angela Bassett, Loretta Devi…  [Forest Whitaker]
4  [Steve Martin, Diane Keaton, Martin Short, Kim…    [Charles Shyer]

      id                                         keywords
0    862  [jealousy, toy, boy, friendship, friends, riva…
1   8844  [board game, disappearance, based on children'…
2  15602  [fishing, best friend, duringcreditsstinger, o…
3  31357  [based on novel, interracial relationship, sin…
4  11862  [baby, midlife crisis, confidence, aging, daug…
```

[14]:
```
movies_df['id']=movies_df['id'].astype(int).astype("int64")

movies_df = movies_df.merge(credits_df)
```

[15]:
```
# Some columns have values of bad type, for example strings in 'budget'.
movies_df['budget'] = pd.to_numeric(movies_df['budget'], errors='coerce')
movies_df['revenue'] = pd.to_numeric(movies_df['revenue'], errors='coerce')
movies_df = movies_df.dropna()
```

[16]:
```
movies_df.head(10)
```

[16]:
```
     budget                            genres     id  \
0  30000000         [Animation, Comedy, Family]    862
1  65000000        [Adventure, Fantasy, Family]   8844
2         0                   [Romance, Comedy]  15602
3  16000000            [Comedy, Drama, Romance]  31357
4         0                            [Comedy]  11862
5  60000000    [Action, Crime, Drama, Thriller]    949
6  58000000                   [Comedy, Romance]  11860
7         0  [Action, Adventure, Drama, Family]  45325
8  35000000       [Action, Adventure, Thriller]   9091
9  58000000       [Adventure, Action, Thriller]    710

                original_title  \
0                    Toy Story
1                      Jumanji
2             Grumpier Old Men
3            Waiting to Exhale
4  Father of the Bride Part II
5                         Heat
6                      Sabrina
```

```
7                 Tom and Huck
8                 Sudden Death
9                    GoldenEye

                               production_companies  \
0                          [Pixar Animation Studios]
1  [TriStar Pictures, Teitler Film, Interscope Co…
2                     [Warner Bros., Lancaster Gate]
3           [Twentieth Century Fox Film Corporation]
4       [Sandollar Productions, Touchstone Pictures]
5  [Regency Enterprises, Forward Pass, Warner Bros.]
6  [Paramount Pictures, Scott Rudin Productions, …
7                            [Walt Disney Pictures]
8  [Universal Pictures, Imperial Entertainment, S…
9                  [United Artists, Eon Productions]

                       production_countries release_date      revenue  \
0                [United States of America]   1995-10-30  373554033.0
1                [United States of America]   1995-12-15  262797249.0
2                [United States of America]   1995-12-22          0.0
3                [United States of America]   1995-12-22   81452156.0
4                [United States of America]   1995-02-10   76578911.0
5                [United States of America]   1995-12-15  187436818.0
6         [Germany, United States of America]   1995-12-15          0.0
7                [United States of America]   1995-12-22          0.0
8                [United States of America]   1995-12-22   64350171.0
9  [United Kingdom, United States of America]   1995-11-16  352194034.0

                       title  \
0                  Toy Story
1                    Jumanji
2           Grumpier Old Men
3          Waiting to Exhale
4  Father of the Bride Part II
5                       Heat
6                    Sabrina
7               Tom and Huck
8               Sudden Death
9                  GoldenEye

                                              cast           director  \
0  [Tom Hanks, Tim Allen, Don Rickles, Jim Varney…    [John Lasseter]
1  [Robin Williams, Jonathan Hyde, Kirsten Dunst,…     [Joe Johnston]
2  [Walter Matthau, Jack Lemmon, Ann-Margret, Sop…    [Howard Deutch]
3  [Whitney Houston, Angela Bassett, Loretta Devi…  [Forest Whitaker]
4  [Steve Martin, Diane Keaton, Martin Short, Kim…    [Charles Shyer]
5  [Al Pacino, Robert De Niro, Val Kilmer, Jon Vo…     [Michael Mann]
```

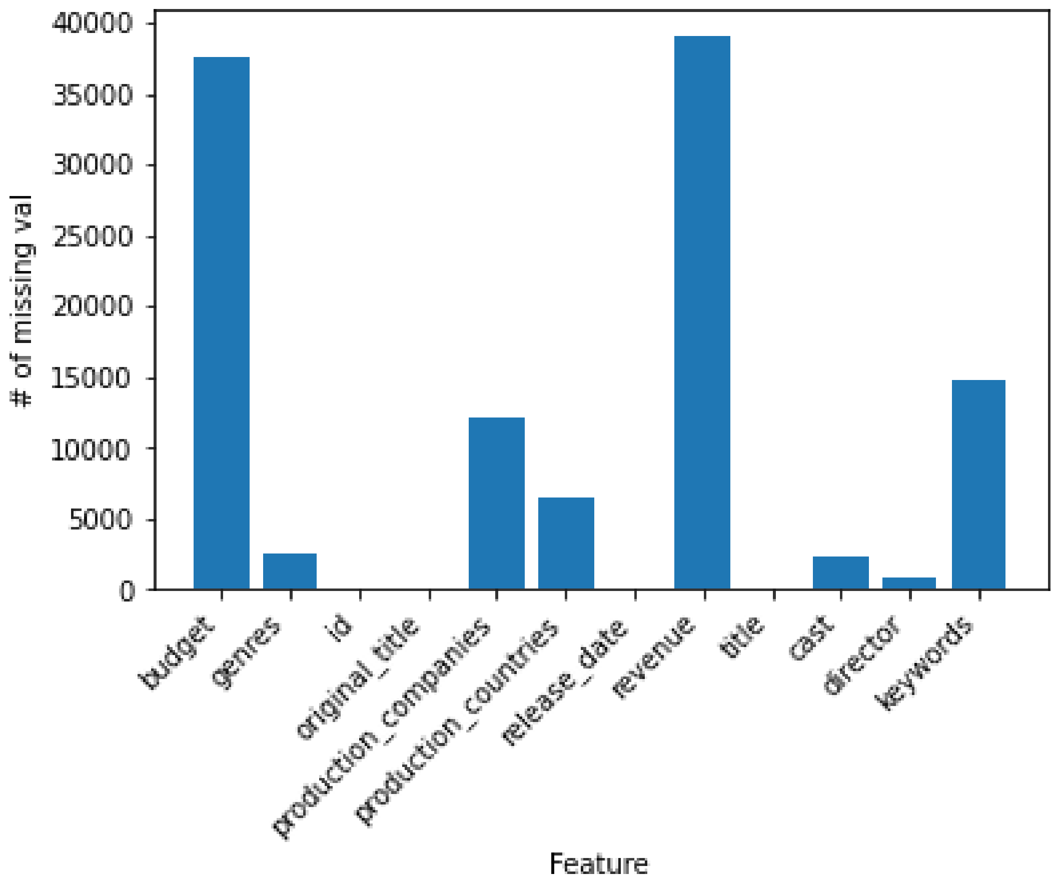
```
6  [Harrison Ford, Julia Ormond, Greg Kinnear, An…   [Sydney Pollack]
7  [Jonathan Taylor Thomas, Brad Renfro, Rachael …     [Peter Hewitt]
8  [Jean-Claude Van Damme, Powers Boothe, Dorian …      [Peter Hyams]
9  [Pierce Brosnan, Sean Bean, Izabella Scorupco,…  [Martin Campbell]

                                            keywords
0  [jealousy, toy, boy, friendship, friends, riva…
1  [board game, disappearance, based on children'…
2  [fishing, best friend, duringcreditsstinger, o…
3  [based on novel, interracial relationship, sin…
4  [baby, midlife crisis, confidence, aging, daug…
5  [robbery, detective, bank, obsession, chase, s…
6  [paris, brother brother relationship, chauffeu…
7                                                 []
8    [terrorist, hostage, explosive, vice president]
9  [cuba, falsely accused, secret identity, compu…
```

[26]:

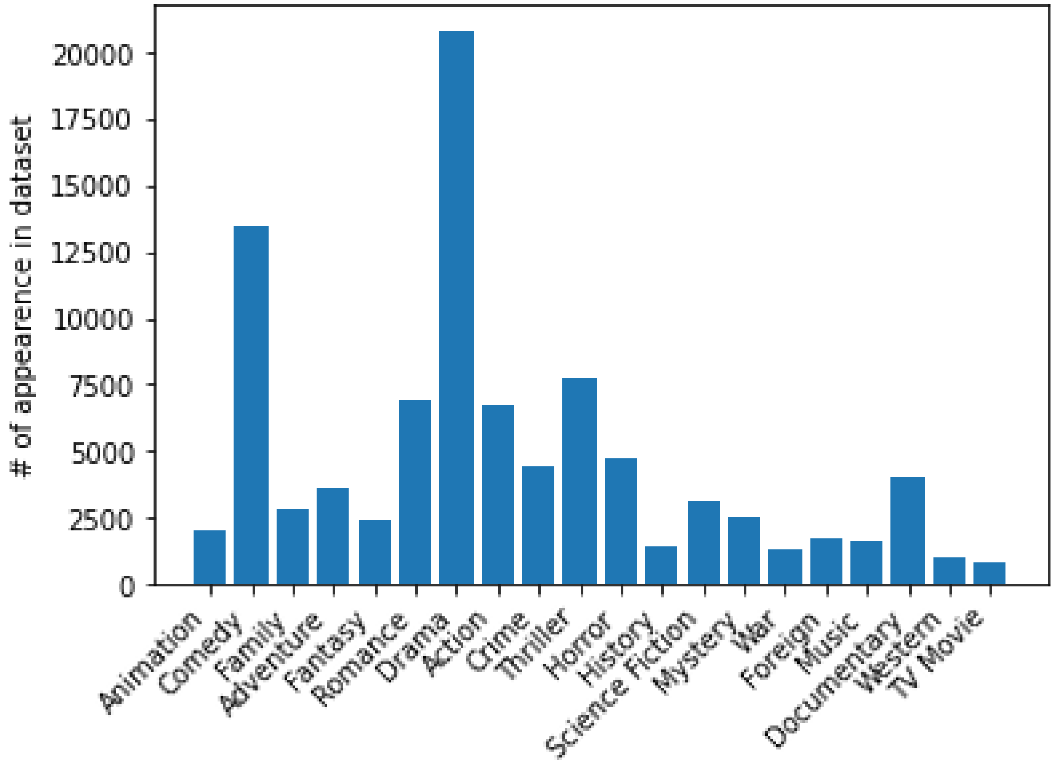
```python
#data statistics
sum_missing_val = []
for col in movies_df.columns:
    sum = 0
    for x in movies_df[col]:
        if x == [] or x == 0 or x == None:
            sum += 1
    sum_missing_val.append(sum)


x = movies_df.columns
num_col = len(x)
y = sum_missing_val
plt.bar(x,y)
plt.xlabel('Feature')
plt.ylabel('# of missing val')
plt.xticks(rotation=45, ha="right")
plt.show()
```



[36]:
```python
genres = dd(int)
for x in movies_df['genres']:
    for genre in x:
        genres[genre] += 1

genres_names = genres.keys()
num_genres = len(genres_names)
y = genres.values()
plt.bar(genres_names,y)
plt.ylabel('# of appearence in dataset')
plt.xticks(rotation=45, ha="right")
plt.show()
```



[56]:

```
# which genres appear with others the most

genres = [dd(int) for x in genres_names]
for x in movies_df['genres']:
    for genre1 in x:
        for genre2 in x:
            genres[list(genres_names).index(genre1)][genre2] +=1

for genre in list(genres_names):

    most_common_pair_genre = genres[list(genres_names).index(genre)].keys()
    most_common_pair_count = genres[list(genres_names).index(genre)].values()
    most_common_pairs = list(zip(most_common_pair_count,most_common_pair_genre))
    most_common_pairs = sorted(most_common_pairs,reverse=True)
    most_common_pairs = most_common_pairs[1:3]
    print(genre,": ", most_common_pairs)
```

```
Animation :  [(878, 'Family'), (564, 'Comedy')]
Comedy :  [(4295, 'Drama'), (3197, 'Romance')]
Family :  [(1205, 'Comedy'), (878, 'Animation')]
Adventure :  [(1775, 'Action'), (1071, 'Drama')]
Fantasy :  [(744, 'Comedy'), (712, 'Drama')]
```

```
Romance :  [(4605, 'Drama'), (3197, 'Comedy')]
Drama :  [(4605, 'Romance'), (4295, 'Comedy')]
Action :  [(2395, 'Thriller'), (2368, 'Drama')]
Crime :  [(2581, 'Drama'), (2055, 'Thriller')]
Thriller :  [(3491, 'Drama'), (2395, 'Action')]
Horror :  [(1951, 'Thriller'), (890, 'Science Fiction')]
History :  [(1098, 'Drama'), (346, 'War')]
Science Fiction :  [(1101, 'Action'), (890, 'Horror')]
Mystery :  [(1535, 'Thriller'), (1201, 'Drama')]
War :  [(1006, 'Drama'), (346, 'History')]
Foreign :  [(1009, 'Drama'), (408, 'Comedy')]
Music :  [(645, 'Drama'), (578, 'Comedy')]
Documentary :  [(348, 'Music'), (216, 'Drama')]
Western :  [(371, 'Action'), (277, 'Drama')]
TV Movie :  [(406, 'Drama'), (174, 'Comedy')]
```

[2]:
```
original_movies_df = movies_df.copy() # Useful to go back to the original␣
↪datframe
movies_df.head()
```

[2]:
```
     budget                       genres     id              original_title  \
0  30000000   [Animation, Comedy, Family]    862                   Toy Story
1  65000000  [Adventure, Fantasy, Family]   8844                     Jumanji
2         0             [Romance, Comedy]  15602            Grumpier Old Men
3  16000000      [Comedy, Drama, Romance]  31357           Waiting to Exhale
4         0                      [Comedy]  11862  Father of the Bride Part II

                                production_companies  \
0                            [Pixar Animation Studios]
1  [TriStar Pictures, Teitler Film, Interscope Co…
2                       [Warner Bros., Lancaster Gate]
3             [Twentieth Century Fox Film Corporation]
4          [Sandollar Productions, Touchstone Pictures]

        production_countries release_date      revenue  \
0  [United States of America]   1995-10-30  373554033.0
1  [United States of America]   1995-12-15  262797249.0
2  [United States of America]   1995-12-22          0.0
3  [United States of America]   1995-12-22   81452156.0
4  [United States of America]   1995-02-10   76578911.0

                         title  \
0                    Toy Story
1                      Jumanji
2             Grumpier Old Men
3            Waiting to Exhale
4  Father of the Bride Part II

                                                cast            director  \
0  [Tom Hanks, Tim Allen, Don Rickles, Jim Varney…    [John Lasseter]
1  [Robin Williams, Jonathan Hyde, Kirsten Dunst,…     [Joe Johnston]
2  [Walter Matthau, Jack Lemmon, Ann-Margret, Sop…    [Howard Deutch]
3  [Whitney Houston, Angela Bassett, Loretta Devi…  [Forest Whitaker]
4  [Steve Martin, Diane Keaton, Martin Short, Kim…    [Charles Shyer]

                                            keywords
0  [jealousy, toy, boy, friendship, friends, riva…
1  [board game, disappearance, based on children'…
2  [fishing, best friend, duringcreditsstinger, o…
3  [based on novel, interracial relationship, sin…
4  [baby, midlife crisis, confidence, aging, daug…
```

[3]:

```python
from sklearn.multioutput import MultiOutputClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
import xgboost as xgb
import gensim.downloader
import numpy as np
```

[4]:

```python
# Replaces column col of lists of labels with a binary matrix.
# Returns new dataframe and new columns' names.
def binarize_column(df, col):
    mlb = MultiLabelBinarizer()
    return df.join(pd.DataFrame(mlb.fit_transform(df[col]), columns=mlb.
 ↪classes_, index=df.index),
                   rsuffix="_suffix").drop(col, axis=1), list(mlb.classes_)

# Takes pandas Series with lists of labels as values.
# Returns a list of labels for which number of occurances > limit.
# Number of labels for 'keywords': > 0 : 18186, > 10 : 2193, > 20 : 1158
def popular_labels(series, limit):
    counts = {}
    for l in series:
        for v in l:
            counts[v] = counts.get(v, 0)+1
    return [k for k,v in counts.items() if v > limit]

# Takes pandas Series with lists of labels as values.
# Creates a new series with only labels that occur at least 'limit' times.
def limit_labels(series, limit):
    new_series = series.copy()
    labels = popular_labels(series, limit)
    for i, l in series.iteritems():
        new_l = [val for val in l if val in labels]
        new_series[i] = new_l
    return new_series
```

[5]:

```python
# Transform categorical features into binary matrixes
movies_df['production_countries'] =␣
 ↪limit_labels(movies_df['production_countries'], 20)
movies_df, _ = binarize_column(movies_df, 'production_countries')
movies_df['keywords'] = limit_labels(movies_df['keywords'], 20)
movies_df, _ = binarize_column(movies_df, 'keywords')
```

```python
movies_df.head()
```

[5]:

```
      budget                          genres     id               original_title  \
0  30000000   [Animation, Comedy, Family]    862                    Toy Story
1  65000000  [Adventure, Fantasy, Family]   8844                      Jumanji
2         0             [Romance, Comedy]  15602             Grumpier Old Men
3  16000000      [Comedy, Drama, Romance]  31357            Waiting to Exhale
4         0                      [Comedy]  11862  Father of the Bride Part II

                              production_companies release_date  \
0                          [Pixar Animation Studios]   1995-10-30
1  [TriStar Pictures, Teitler Film, Interscope Co…   1995-12-15
2                     [Warner Bros., Lancaster Gate]   1995-12-22
3           [Twentieth Century Fox Film Corporation]   1995-12-22
4       [Sandollar Productions, Touchstone Pictures]   1995-02-10

      revenue                        title  \
0  373554033.0                    Toy Story
1  262797249.0                      Jumanji
2          0.0             Grumpier Old Men
3   81452156.0            Waiting to Exhale
4   76578911.0  Father of the Bride Part II

                                                cast            director  …  \
0  [Tom Hanks, Tim Allen, Don Rickles, Jim Varney…    [John Lasseter]  …
1  [Robin Williams, Jonathan Hyde, Kirsten Dunst,…     [Joe Johnston]  …
2  [Walter Matthau, Jack Lemmon, Ann-Margret, Sop…    [Howard Deutch]  …
3  [Whitney Houston, Angela Bassett, Loretta Devi…  [Forest Whitaker]  …
4  [Steve Martin, Diane Keaton, Martin Short, Kim…    [Charles Shyer]  …

   yacht  yakuza  young adult  young boy  young love  youth  zatoichi  zombie  \
0      0       0            0          0           0      0         0       0
1      0       0            0          0           0      0         0       0
2      0       0            0          0           0      0         0       0
3      0       0            0          0           0      0         0       0
4      0       0            0          0           0      0         0       0

   zombie apocalypse  zoo
0                  0    0
1                  0    0
2                  0    0
3                  0    0
4                  0    0

[5 rows x 1530 columns]
```

[6]:
```
#create list of possible genres
genres = []
for x in movies_df['genres']:
    for genre in x:
        if genre not in genres:
            genres.append(genre)
```

[7]:
```
train_df, test_df = train_test_split(movies_df, test_size=0.2)

X_cols = ['budget', 'revenue'] # For now only numeric features
X_train = train_df[X_cols]
X_test = test_df[X_cols]

# We need to transform 'genres' since this is multi-label classification
mlb = MultiLabelBinarizer()

y_train = mlb.fit_transform(train_df['genres'])
y_test = mlb.fit_transform(test_df['genres'])
```

[8]:
```
def random_forest(X_train,y_train,X_test,y_test):
    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    y_pred = rfc.predict(X_test)
    return classification_report(y_test ,y_pred,target_names=genres,␣
 ↪output_dict=True)

def xgboost(X_train,y_train,X_test,y_test):
    xgbc = MultiOutputClassifier(xgb.XGBClassifier(verbosity = 0))
    xgbc.fit(X_train, y_train)
    y_pred = xgbc.predict(X_test)
    return classification_report(y_test ,y_pred,target_names=genres,␣
 ↪output_dict=True)

def logistic_reg(X_train, y_train,X_test,y_test):
    clf = MultiOutputClassifier(LogisticRegression()).fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    return classification_report(y_test ,y_pred,target_names=genres,␣
 ↪output_dict=True)
```

[9]:
```
# Random forest, only numeric columns

random_forest(X_train, y_train,X_test,y_test)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
```

```
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[9]: {'Animation': {'precision': 0.32171581769436997,
       'recall': 0.09230769230769231,
       'f1-score': 0.1434548714883443,
       'support': 1300},
      'Comedy': {'precision': 0.3448275862068966,
       'recall': 0.09722222222222222,
       'f1-score': 0.1516793066088841,
       'support': 720},
      'Family': {'precision': 0.13636363636363635,
       'recall': 0.01485148514851485,
       'f1-score': 0.026785714285714284,
       'support': 404},
      'Adventure': {'precision': 0.3697632058287796,
       'recall': 0.07645951035781544,
       'f1-score': 0.12671660424469414,
       'support': 2655},
      'Fantasy': {'precision': 0.17801047120418848,
       'recall': 0.03711790393013101,
       'f1-score': 0.06142728093947606,
       'support': 916},
      'Romance': {'precision': 0.14285714285714285,
       'recall': 0.007537688442211055,
       'f1-score': 0.014319809069212411,
       'support': 796},
      'Drama': {'precision': 0.5537848605577689,
       'recall': 0.13384689455946075,
       'f1-score': 0.2155874369910818,
       'support': 4154},
      'Action': {'precision': 0.2222222222222222,
       'recall': 0.03103448275862069,
       'f1-score': 0.05446293494704992,
       'support': 580},
      'Crime': {'precision': 0.19540229885057472,
       'recall': 0.03736263736263736,
       'f1-score': 0.06273062730627306,
       'support': 455},
      'Thriller': {'precision': 0.15789473684210525,
       'recall': 0.009174311926605505,
       'f1-score': 0.017341040462427747,
```

```
 'support': 327},
'Horror': {'precision': 0.22857142857142856,
 'recall': 0.02877697841726619,
 'f1-score': 0.051118210862619806,
 'support': 278},
'History': {'precision': 0.22627737226277372,
 'recall': 0.033155080213903745,
 'f1-score': 0.057835820895522395,
 'support': 935},
'Science Fiction': {'precision': 0.08823529411764706,
 'recall': 0.008823529411764706,
 'f1-score': 0.0160427807486631,
 'support': 340},
'Mystery': {'precision': 0.0641025641025641,
 'recall': 0.0102880658436214,
 'f1-score': 0.01773049645390071,
 'support': 486},
'War': {'precision': 0.23293172690763053,
 'recall': 0.041105598866052445,
 'f1-score': 0.06987951807228915,
 'support': 1411},
'Foreign': {'precision': 0.1523809523809524,
 'recall': 0.02622950819672131,
 'f1-score': 0.04475524475524475,
 'support': 610},
'Music': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 164},
'Documentary': {'precision': 0.2558746736292428,
 'recall': 0.0625,
 'f1-score': 0.10046130189646335,
 'support': 1568},
'Western': {'precision': 0.23809523809523808,
 'recall': 0.01858736059479554,
 'f1-score': 0.034482758620689655,
 'support': 269},
'TV Movie': {'precision': 0.08333333333333333,
 'recall': 0.0044444444444444444,
 'f1-score': 0.008438818565400843,
 'support': 225},
'micro avg': {'precision': 0.34484649122807015,
 'recall': 0.06765987199483676,
 'f1-score': 0.11312440987365677,
 'support': 18593},
'macro avg': {'precision': 0.20963222810142473,
 'recall': 0.03854126975022405,
 'f1-score': 0.06376252886069758,
 'support': 18593},
'weighted avg': {'precision': 0.31150624153311707,
```

```
  'recall': 0.06765987199483676,
  'f1-score': 0.11019897885702132,
  'support': 18593},
 'samples avg': {'precision': 0.0739740008594757,
  'recall': 0.0581297194425686,
  'f1-score': 0.05943915643636313,
  'support': 18593}}
```

[10]:
```
# Logistic regression, only numeric columns
logistic_reg(X_train, y_train,X_test,y_test)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: 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))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[10]:
```
{'Animation': {'precision': 0.0,
  'recall': 0.0,
  'f1-score': 0.0,
  'support': 1300},
 'Comedy': {'precision': 0.09836065573770492,
  'recall': 0.075,
  'f1-score': 0.08510638297872339,
  'support': 720},
 'Family': {'precision': 0.04241071428571429,
  'recall': 0.04702970297029703,
  'f1-score': 0.04460093896713615,
  'support': 404},
 'Adventure': {'precision': 0.33760683760683763,
  'recall': 0.05951035781544256,
  'f1-score': 0.10118475824527698,
  'support': 2655},
 'Fantasy': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 916},
 'Romance': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 796},
 'Drama': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4154},
 'Action': {'precision': 0.06016597510373444,
```

```
 'recall': 0.05,
 'f1-score': 0.054613935969868174,
 'support': 580},
'Crime': {'precision': 0.060215053763440864,
 'recall': 0.06153846153846154,
 'f1-score': 0.060869565217391314,
 'support': 455},
'Thriller': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 327},
'Horror': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 278},
'History': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 935},
'Science Fiction': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 340},
'Mystery': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 486},
'War': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1411},
'Foreign': {'precision': 0.0625,
 'recall': 0.04426229508196721,
 'f1-score': 0.051823416506717845,
 'support': 610},
'Music': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 164},
'Documentary': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 1568},
'Western': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 269},
'TV Movie': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 225},
'micro avg': {'precision': 0.11075949367088607,
 'recall': 0.016941859839724627,
 'f1-score': 0.02938844054671829,
 'support': 18593},
'macro avg': {'precision': 0.0330629618248716,
 'recall': 0.016867040870308415,
 'f1-score': 0.019909949894255694,
 'support': 18593},
'weighted avg': {'precision': 0.05834017477390747 4,
 'recall': 0.016941859839724627,
 'f1-score': 0.02360699872716338,
 'support': 18593},
'samples avg': {'precision': 0.0064281621544191375,
 'recall': 0.014881463973642746,
```

```
 'f1-score': 0.008117270687103087,
 'support': 18593}}
```

[11]:
```
X_cols_to_drop = ['genres', 'id', 'original_title', 'production_companies',
                  'release_date', 'title', 'cast', 'director']
X_train = train_df.drop(columns=X_cols_to_drop, axis=1)
X_test = test_df.drop(columns=X_cols_to_drop, axis=1)
```

[12]:
```
# Random forest with limited keywords and production_countries
random_forest(X_train,y_train,X_test,y_test)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[12]:
```
{'Animation': {'precision': 0.5884413309982487,
  'recall': 0.25846153846153846,
  'f1-score': 0.35916622127204706,
  'support': 1300},
 'Comedy': {'precision': 0.5869565217391305,
  'recall': 0.15,
  'f1-score': 0.23893805309734512,
  'support': 720},
 'Family': {'precision': 0.7064220183486238,
  'recall': 0.1905940594059406,
  'f1-score': 0.3001949317738792,
  'support': 404},
 'Adventure': {'precision': 0.5570739549839229,
  'recall': 0.26101694915254237,
  'f1-score': 0.355475763016158,
  'support': 2655},
 'Fantasy': {'precision': 0.4897260273972603,
  'recall': 0.15611353711790393,
  'f1-score': 0.23675496688741723,
  'support': 916},
 'Romance': {'precision': 0.5233160621761658,
  'recall': 0.12688442211055276,
  'f1-score': 0.20424671385237614,
  'support': 796},
 'Drama': {'precision': 0.653556211078335,
```

```
 'recall': 0.48001925854597977,
 'f1-score': 0.5535045107564192,
 'support': 4154},
'Action': {'precision': 0.46099290780141844,
 'recall': 0.11206896551724138,
 'f1-score': 0.18030513176144242,
 'support': 580},
'Crime': {'precision': 0.55,
 'recall': 0.12087912087912088,
 'f1-score': 0.19819819819819817,
 'support': 455},
'Thriller': {'precision': 0.3409090909090909,
 'recall': 0.045871559633027525,
 'f1-score': 0.08086253369272238,
 'support': 327},
'Horror': {'precision': 0.3684210526315789,
 'recall': 0.050359712230215826,
 'f1-score': 0.08860759493670885,
 'support': 278},
'History': {'precision': 0.7518987341772152,
 'recall': 0.3176470588235294,
 'f1-score': 0.4466165413533834,
 'support': 935},
'Science Fiction': {'precision': 0.5363636363636364,
 'recall': 0.17352941176470588,
 'f1-score': 0.2622222222222222,
 'support': 340},
'Mystery': {'precision': 0.411214953271028,
 'recall': 0.09053497942386832,
 'f1-score': 0.14839797639123103,
 'support': 486},
'War': {'precision': 0.48058252427184467,
 'recall': 0.1403260099220411,
 'f1-score': 0.21722435545803617,
 'support': 1411},
'Foreign': {'precision': 0.7184873949579832,
 'recall': 0.28032786885245903,
 'f1-score': 0.4033018867924528,
 'support': 610},
'Music': {'precision': 0.32558139534883723,
 'recall': 0.08536585365853659,
 'f1-score': 0.1352657004830918,
 'support': 164},
'Documentary': {'precision': 0.5618556701030928,
 'recall': 0.20854591836734693,
 'f1-score': 0.3041860465116279,
 'support': 1568},
```

```
 'Western': {'precision': 0.5555555555555556,
  'recall': 0.16728624535315986,
  'f1-score': 0.2571428571428572,
  'support': 269},
 'TV Movie': {'precision': 0.75,
  'recall': 0.36,
  'f1-score': 0.48648648648648657,
  'support': 225},
 'micro avg': {'precision': 0.6013925152306353,
  'recall': 0.26015166998332706,
  'f1-score': 0.36319267157230817,
  'support': 18593},
 'macro avg': {'precision': 0.5458677521056484,
  'recall': 0.18879162346098552,
  'f1-score': 0.27285493460430515,
  'support': 18593},
 'weighted avg': {'precision': 0.5764401531716534,
  'recall': 0.26015166998332706,
  'f1-score': 0.34624404430698164,
  'support': 18593},
 'samples avg': {'precision': 0.3603441484028076,
  'recall': 0.26288753146295046,
  'f1-score': 0.28425871924475277,
  'support': 18593}}
```

[13]:
```
# Logistic regression with limited keywords and production_countries
X_train.head()
logistic_reg(X_train,y_train,X_test,y_test)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: 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))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[13]:
```
{'Animation': {'precision': 0.0,
  'recall': 0.0,
```

```
 'f1-score': 0.0,
 'support': 1300},
'Comedy': {'precision': 0.09836065573770492,
 'recall': 0.075,
 'f1-score': 0.08510638297872339,
 'support': 720},
'Family': {'precision': 0.04241071428571429,
 'recall': 0.04702970297029703,
 'f1-score': 0.04460093896713615,
 'support': 404},
'Adventure': {'precision': 0.33760683760683763,
 'recall': 0.05951035781544256,
 'f1-score': 0.10118475824527698,
 'support': 2655},
'Fantasy': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 916},
'Romance': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 796},
'Drama': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4154},
'Action': {'precision': 0.06016597510373444,
 'recall': 0.05,
 'f1-score': 0.054613935969868174,
 'support': 580},
'Crime': {'precision': 0.060215053763440864,
 'recall': 0.06153846153846154,
 'f1-score': 0.060869565217391314,
 'support': 455},
'Thriller': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 327},
'Horror': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 278},
'History': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 935},
'Science Fiction': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 340},
'Mystery': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 486},
'War': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1411},
'Foreign': {'precision': 0.0625,
 'recall': 0.04426229508196721,
 'f1-score': 0.051823416506717845,
 'support': 610},
'Music': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 164},
'Documentary': {'precision': 0.0,
 'recall': 0.0,
 'f1-score': 0.0,
 'support': 1568},
'Western': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 269},
```

```
 'TV Movie': {'precision': 0.0,
  'recall': 0.0,
  'f1-score': 0.0,
  'support': 225},
 'micro avg': {'precision': 0.11075949367088607,
  'recall': 0.016941859839724627,
  'f1-score': 0.02938844054671829,
  'support': 18593},
 'macro avg': {'precision': 0.0330629618248716,
  'recall': 0.016867040870308415,
  'f1-score': 0.019909949894255694,
  'support': 18593},
 'weighted avg': {'precision': 0.058340174773907474,
  'recall': 0.016941859839724627,
  'f1-score': 0.02360699872716338,
  'support': 18593},
 'samples avg': {'precision': 0.0064281621544191375,
  'recall': 0.014881463973642746,
  'f1-score': 0.008117270687103087,
  'support': 18593}}
```

[14]:
```
# XGBoost with limited keywords and production_countries
xgboost(X_train,y_train,X_test,y_test)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[14]:
```
{'Animation': {'precision': 0.7453271028037384,
  'recall': 0.2453846153846154,
  'f1-score': 0.36921296296296297,
  'support': 1300},
 'Comedy': {'precision': 0.6994219653179191,
```

```
 'recall': 0.16805555555555557,
 'f1-score': 0.27099664053751404,
 'support': 720},
'Family': {'precision': 0.8706896551724138,
 'recall': 0.25,
 'f1-score': 0.3884615384615385,
 'support': 404},
'Adventure': {'precision': 0.7602040816326531,
 'recall': 0.16836158192090395,
 'f1-score': 0.2756706753006475,
 'support': 2655},
'Fantasy': {'precision': 0.6260504201680672,
 'recall': 0.16266375545851527,
 'f1-score': 0.2582322357019064,
 'support': 916},
'Romance': {'precision': 0.7613636363636364,
 'recall': 0.08417085427135679,
 'f1-score': 0.15158371040723984,
 'support': 796},
'Drama': {'precision': 0.6886861313868613,
 'recall': 0.454260953298026,
 'f1-score': 0.5474325500435161,
 'support': 4154},
'Action': {'precision': 0.7037037037037037,
 'recall': 0.1310344827586207,
 'f1-score': 0.22093023255813954,
 'support': 580},
'Crime': {'precision': 0.6464646464646465,
 'recall': 0.14065934065934066,
 'f1-score': 0.23104693140794225,
 'support': 455},
'Thriller': {'precision': 0.5,
 'recall': 0.0030581039755351682,
 'f1-score': 0.0060790273556231,
 'support': 327},
'Horror': {'precision': 0.5476190476190477,
 'recall': 0.08273381294964029,
 'f1-score': 0.14375,
 'support': 278},
'History': {'precision': 0.8663239074550129,
 'recall': 0.360427807486631,
 'f1-score': 0.5090634441087614,
 'support': 935},
'Science Fiction': {'precision': 0.616,
 'recall': 0.22647058823529412,
 'f1-score': 0.3311827956989247,
 'support': 340},
```

```
 'Mystery': {'precision': 0.5138888888888888,
  'recall': 0.07613168724279835,
  'f1-score': 0.13261648745519714,
  'support': 486},
 'War': {'precision': 0.6723404255319149,
  'recall': 0.11197732104890148,
  'f1-score': 0.19198055893074117,
  'support': 1411},
 'Foreign': {'precision': 0.8392857142857143,
  'recall': 0.38524590163934425,
  'f1-score': 0.5280898876404494,
  'support': 610},
 'Music': {'precision': 0.5384615384615384,
  'recall': 0.042682926829268296,
  'f1-score': 0.0790960451977401,
  'support': 164},
 'Documentary': {'precision': 0.7331730769230769,
  'recall': 0.19451530612244897,
  'f1-score': 0.3074596774193548,
  'support': 1568},
 'Western': {'precision': 0.7162162162162162,
  'recall': 0.1970260223048327,
  'f1-score': 0.30903790087463556,
  'support': 269},
 'TV Movie': {'precision': 0.8640776699029126,
  'recall': 0.39555555555555555,
  'f1-score': 0.5426829268292682,
  'support': 225},
 'micro avg': {'precision': 0.7193869489650814,
  'recall': 0.24487710428655945,
  'f1-score': 0.3653799855549314,
  'support': 18593},
 'macro avg': {'precision': 0.695464891414898,
  'recall': 0.19402080863485924,
  'f1-score': 0.28973031144460515,
  'support': 18593},
 'weighted avg': {'precision': 0.7129451507153145,
  'recall': 0.24487710428655945,
  'f1-score': 0.34511600094984046,
  'support': 18593},
 'samples avg': {'precision': 0.3529777252542615,
  'recall': 0.2471959604641169,
  'f1-score': 0.273300101512392,
  'support': 18593}}
```

[15]:
```
# Use pre-trained model based on Wikipedia 2014 + Gigaword (https://nlp.
↪stanford.edu/projects/glove/)
```

```
model = gensim.downloader.load('glove-wiki-gigaword-50')
```

[16]:
```
# Takes a string and returns the average of its words vectors.
def string_to_vector(phrase):
    phrase = phrase.lower()
    phrase = ''.join([c for c in phrase if c.isalnum() or c == ' '])
    vectors = np.array([model[word] for word in phrase.split() if word in␣
 ↪model])
    if len(vectors) == 0:
        return np.full(50, np.nan)
    return np.average(vectors, axis=0)

# Takes a list of strings and returns the average of its words vectors.
def string_list_to_vector(phrase_list):
    vector_list = [string_to_vector(phrase) for phrase in phrase_list]
    vector_array = np.array([v for v in vector_list if not np.any(np.
 ↪isnan(v))]) # skip nans
    if len(vector_array) == 0:
        return np.full(50, np.nan)
    return np.average(vector_array, axis=0)

# Replaces column col of strings / lists of strings with a column of word␣
 ↪vectors.
def string_column_to_vector_column(df, col_name, new_names):
    col = df[col_name]
    if type(col[0]) == str:
        fun = string_to_vector
    else:
        fun = string_list_to_vector
    array = np.array([fun(row) for row in col])
    return df.join(pd.DataFrame(array, columns=new_names, index=df.index),
                   rsuffix="_suffix").drop(col_name, axis=1)
```

[17]:
```
movies_df = original_movies_df.copy() # Reverse all changes to the dataframe

title_cols = ['title'+str(i) for i in range(50)]
keywords_cols = ['keywords'+str(i) for i in range(50)]

# Drop rows with empty genres. Maybe we should do that in data_prep?
movies_df = movies_df[movies_df['genres'].apply(lambda x : x != [])]

# Columns 'title' and 'keywords' are transformed using Word2Vec
movies_df = string_column_to_vector_column(movies_df, 'title', title_cols)
movies_df = string_column_to_vector_column(movies_df, 'keywords', keywords_cols)

# Columns 'production_companies', 'production_countries', 'cast', 'genres'
# are transformed by limiting the labels and using MultiLabelBinarizer.
```

```python
movies_df['production_companies'] =␣
↪limit_labels(movies_df['production_companies'], 20)
movies_df, production_companies_cols = binarize_column(movies_df,␣
↪'production_companies')
movies_df['production_countries'] =␣
↪limit_labels(movies_df['production_countries'], 20)
movies_df, production_countries_cols = binarize_column(movies_df,␣
↪'production_countries')
movies_df['cast'] = limit_labels(movies_df['cast'], 50)
movies_df, cast_cols = binarize_column(movies_df, 'cast')

movies_df['genres'] = limit_labels(movies_df['genres'], 50)
movies_df, genres_cols = binarize_column(movies_df, 'genres')

# TODO: Transform the column 'director' - it is not multilabel.
movies_df.head()
```

[17]:

```
     budget     id               original_title release_date      revenue  \
0  30000000    862                    Toy Story   1995-10-30  373554033.0
1  65000000   8844                      Jumanji   1995-12-15  262797249.0
2         0  15602             Grumpier Old Men   1995-12-22          0.0
3  16000000  31357            Waiting to Exhale   1995-12-22   81452156.0
4         0  11862  Father of the Bride Part II   1995-02-10   76578911.0

             director    title0    title1    title2    title3  …  History  \
0    [John Lasseter]  0.158795 -0.067820 -0.065135  0.133620  …        0
1     [Joe Johnston] -0.025142 -0.792810 -0.613600 -0.107130  …        0
2    [Howard Deutch] -0.612903  0.785603 -0.008300 -0.742430  …        0
3  [Forest Whitaker]  0.583170 -0.090578  0.333530 -0.928237  …        0
4    [Charles Shyer]  0.581018  0.635348 -0.503253 -0.341070  …        0

   Horror  Music  Mystery  Romance  Science Fiction  TV Movie  Thriller  War  \
0       0      0        0        0                0         0         0    0
1       0      0        0        0                0         0         0    0
2       0      0        0        1                0         0         0    0
3       0      0        0        1                0         0         0    0
4       0      0        0        0                0         0         0    0

   Western
0        0
1        0
2        0
3        0
4        0

[5 rows x 964 columns]
```

[18]:
```
# Names of the columns that we consider.
# I have only included 'budget', 'revenue', 'title' and 'keywords', as␣
 ↪otherwise it takes a long time to compute.
cols = ['budget', 'revenue']+title_cols+keywords_cols

# Impute the missing values in 'title' and 'keywords'.
# The missing values in the rest of the columns will not change, as they are␣
 ↪not equal to np.nan.

#finding out which strategy for imputation_rder is the best
strategies = ['ascending', 'descending', 'roman', 'arabic', 'random']
micro_avg_rf = []
micro_avg_xg = []
macro_avg_rf = []
macro_avg_xg = []
for s in strategies:
    imp = IterativeImputer(imputation_order=s)
    movies_df = movies_df.drop(cols, axis=1).join(
        pd.DataFrame(imp.fit_transform(movies_df[cols]), columns=imp.
 ↪feature_names_in_, index=movies_df.index))
    train_df, test_df = train_test_split(movies_df, test_size=0.2)
    X_cols = ['budget',␣
 ↪'revenue']+title_cols+keywords_cols+production_companies_cols+production_countries_cols+cas
    X_train = train_df[X_cols]
    X_test = test_df[X_cols]

    y_train = train_df[genres_cols]
    y_test = test_df[genres_cols]

    rf_output = random_forest(X_train,y_train,X_test,y_test)
    xg_output = xgboost(X_train,y_train,X_test,y_test)

    micro_avg_rf.append(rf_output['micro avg'])
    micro_avg_xg.append(xg_output['micro avg'])
    macro_avg_rf.append(rf_output['macro avg'])
    macro_avg_xg.append(xg_output['macro avg'])


print(micro_avg_rf)
print(macro_avg_rf)
print(micro_avg_xg)
print(macro_avg_xg)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
```

```
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
```

```
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[{'precision': 0.7112468407750632, 'recall': 0.18169590013989023, 'f1-score':
0.289448872889346, 'support': 18586}, {'precision': 0.7047560222359481,
'recall': 0.18341102716605048, 'f1-score': 0.2910714285714286, 'support':
18663}, {'precision': 0.7, 'recall': 0.18043944265809217, 'f1-score':
0.2869194716659565, 'support': 18660}, {'precision': 0.7052261929353439,
'recall': 0.1827525293078529, 'f1-score': 0.29028143865317574, 'support':
18681}, {'precision': 0.7141025641025641, 'recall': 0.1788026322829169,
'f1-score': 0.2859954644645073, 'support': 18691}]
[{'precision': 0.8221579025744268, 'recall': 0.08776091651079905, 'f1-score':
0.14214723246187275, 'support': 18586}, {'precision': 0.8324981486922305,
'recall': 0.09190829389339475, 'f1-score': 0.1485989277108873, 'support':
18663}, {'precision': 0.811913148670001, 'recall': 0.08793395123370473,
'f1-score': 0.1414068288911497, 'support': 18660}, {'precision':
0.8182004476311018, 'recall': 0.091694496463873, 'f1-score': 0.1484630409688068,
'support': 18681}, {'precision': 0.8229657397728424, 'recall':
0.08814058236145697, 'f1-score': 0.14349612710304357, 'support': 18691}]
```

```
[{'precision': 0.6693041184824507, 'recall': 0.35499838588184657, 'f1-score':
0.46392912389256086, 'support': 18586}, {'precision': 0.6606379807214549,
'recall': 0.35621282751969136, 'f1-score': 0.4628559493142101, 'support':
18663}, {'precision': 0.6526168597682781, 'recall': 0.3501607717041801,
'f1-score': 0.4557756696428572, 'support': 18660}, {'precision':
0.6595399736548789, 'recall': 0.34842888496333174, 'f1-score':
0.4559719789842382, 'support': 18681}, {'precision': 0.6697987948115617,
'recall': 0.35086405221764483, 'f1-score': 0.46050136928586477, 'support':
18691}]
[{'precision': 0.6934639166631456, 'recall': 0.2590711540995434, 'f1-score':
0.3604746112383656, 'support': 18586}, {'precision': 0.6872905317612212,
'recall': 0.26641458968535975, 'f1-score': 0.36787614224909965, 'support':
18663}, {'precision': 0.6741287885800958, 'recall': 0.25803037266948564,
'f1-score': 0.35716216351698404, 'support': 18660}, {'precision':
0.6816322369064445, 'recall': 0.258732628660318, 'f1-score':
0.35941345947473347, 'support': 18681}, {'precision': 0.6833754318479817,
'recall': 0.25653236528494583, 'f1-score': 0.357858784000396, 'support': 18691}]
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[19]:
```python
print("micro avg rf: ", max(micro_avg_rf,key=lambda x: x['precision']))
print("macro avg rf: ", max(macro_avg_rf,key=lambda x: x['precision']))
print("micro avg xgboost: ", max(micro_avg_xg,key=lambda x: x['precision']))
print("macro avg xgboost: ", max(macro_avg_xg,key=lambda x: x['precision']))
```

```
micro avg rf:  {'precision': 0.7141025641025641, 'recall': 0.1788026322829169,
'f1-score': 0.2859954644645073, 'support': 18691}
macro avg rf:  {'precision': 0.8324981486922305, 'recall': 0.09190829389339475,
'f1-score': 0.1485989277108873, 'support': 18663}
micro avg xgboost:  {'precision': 0.6697987948115617, 'recall':
0.35086405221764483, 'f1-score': 0.46050136928586477, 'support': 18691}
macro avg xgboost:  {'precision': 0.6934639166631456, 'recall':
0.2590711540995434, 'f1-score': 0.3604746112383656, 'support': 18586}
```

[20]:
```python
# Random forest with:
# - keywords and titles processed by Word2Vec
# - 'production_companies_cols', 'production_countries_cols', 'cast' limited␣
 ↪and processed with MultiLabelBinarizer
# - 'budget' and 'revenue'
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print(accuracy_score(y_test, y_pred))
classification_report(y_test ,y_pred,target_names=genres_cols, output_dict=True)
```

```
0.12490073737946682

C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[20]:
```
{'Action': {'precision': 0.8636363636363636,
  'recall': 0.08394698085419734,
  'f1-score': 0.15302013422818792,
  'support': 1358},
 'Adventure': {'precision': 0.8636363636363636,
  'recall': 0.05352112676056338,
  'f1-score': 0.10079575596816975,
  'support': 710},
 'Animation': {'precision': 0.9444444444444444,
  'recall': 0.08717948717948718,
  'f1-score': 0.1596244131455399,
  'support': 390},
 'Comedy': {'precision': 0.757455268389662,
  'recall': 0.14350282485875707,
  'f1-score': 0.24129195693476885,
  'support': 2655},
 'Crime': {'precision': 0.7575757575757576,
  'recall': 0.05263157894736842,
  'f1-score': 0.09842519685039369,
  'support': 950},
 'Documentary': {'precision': 0.8035714285714286,
  'recall': 0.054878048780487805,
  'f1-score': 0.10273972602739724,
  'support': 820},
 'Drama': {'precision': 0.678679588128407,
  'recall': 0.5334444179957153,
  'f1-score': 0.5973610555777689,
  'support': 4201},
 'Family': {'precision': 0.8620689655172413,
  'recall': 0.04638218923933209,
  'f1-score': 0.0880281690140845,
  'support': 539},
 'Fantasy': {'precision': 0.9032258064516129,
  'recall': 0.060215053763440864,
  'f1-score': 0.11290322580645162,
  'support': 465},
 'Foreign': {'precision': 0.7333333333333333,
  'recall': 0.06489675516224189,
  'f1-score': 0.11924119241192412,
  'support': 339},
```

```
'History': {'precision': 0.9375,
 'recall': 0.053003533568904596,
 'f1-score': 0.1003344481605351,
 'support': 283},
'Horror': {'precision': 0.9130434782608695,
 'recall': 0.06702127659574468,
 'f1-score': 0.1248761149653122,
 'support': 940},
'Music': {'precision': 0.65,
 'recall': 0.038461538461538464,
 'f1-score': 0.07262569832402235,
 'support': 338},
'Mystery': {'precision': 0.8235294117647058,
 'recall': 0.051756007393715345,
 'f1-score': 0.0973913043478261,
 'support': 541},
'Romance': {'precision': 0.7840909090909091,
 'recall': 0.05114899925871016,
 'f1-score': 0.09603340292275574,
 'support': 1349},
'Science Fiction': {'precision': 0.8985507246376812,
 'recall': 0.09967845659163987,
 'f1-score': 0.17945007235890015,
 'support': 622},
'TV Movie': {'precision': 1.0,
 'recall': 0.031446540880503145,
 'f1-score': 0.06097560975609756,
 'support': 159},
'Thriller': {'precision': 0.7454545454545455,
 'recall': 0.07889672867222579,
 'f1-score': 0.14269141531322507,
 'support': 1559},
'War': {'precision': 0.7272727272727273,
 'recall': 0.06451612903225806,
 'f1-score': 0.11851851851851852,
 'support': 248},
'Western': {'precision': 1.0,
 'recall': 0.06222222222222222,
 'f1-score': 0.11715481171548117,
 'support': 225},
'micro avg': {'precision': 0.71570492496301,
 'recall': 0.18115670643625273,
 'f1-score': 0.2891298778925796,
 'support': 18691},
'macro avg': {'precision': 0.8323534558083026,
 'recall': 0.08893749481095267,
 'f1-score': 0.1441741111736805,
```

```
 'support': 18691},
'weighted avg': {'precision': 0.7843538588261799,
 'recall': 0.18115670643625273,
 'f1-score': 0.24407303503035138,
 'support': 18691},
'samples avg': {'precision': 0.32061826432217805,
 'recall': 0.21308132782324501,
 'f1-score': 0.24185029126038768,
 'support': 18691}}
```

[21]:
```
# XGBoost with:
# - keywords and titles processed by Word2Vec
# - 'production_companies_cols', 'production_countries_cols', 'cast' limited␣
↪and processed with MultiLabelBinarizer
# - 'budget' and 'revenue'

# Runs about 13 minutes
xgbc = MultiOutputClassifier(xgb.XGBClassifier())
xgbc.fit(X_train, y_train)
y_pred = xgbc.predict(X_test)
print(accuracy_score(y_test, y_pred))
classification_report(y_test ,y_pred,target_names=genres_cols, output_dict=True)
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
0.15507657402155417
```

```
C:\Users\natal\AppData\Local\Programs\Python\Python37\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in samples with no
predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[21]:
```
{'Action': {'precision': 0.7005730659025788,
  'recall': 0.3600883652430044,
  'f1-score': 0.4756809338521401,
  'support': 1358},
 'Adventure': {'precision': 0.6285714285714286,
  'recall': 0.21690140845070421,
  'f1-score': 0.3225130890052356,
  'support': 710},
 'Animation': {'precision': 0.9171974522292994,
```

```
 'recall': 0.36923076923076925,
 'f1-score': 0.526508226691042,
 'support': 390},
'Comedy': {'precision': 0.6428571428571429,
 'recall': 0.36610169491525424,
 'f1-score': 0.4665226781857452,
 'support': 2655},
'Crime': {'precision': 0.6445012787723785,
 'recall': 0.26526315789473687,
 'f1-score': 0.37583892617449666,
 'support': 950},
'Documentary': {'precision': 0.6846153846153846,
 'recall': 0.21707317073170732,
 'f1-score': 0.3296296296296296,
 'support': 820},
'Drama': {'precision': 0.6677026677026677,
 'recall': 0.6136634134729826,
 'f1-score': 0.6395435375837261,
 'support': 4201},
'Family': {'precision': 0.8083832335329342,
 'recall': 0.2504638218923933,
 'f1-score': 0.38243626062322944,
 'support': 539},
'Fantasy': {'precision': 0.6347826086956522,
 'recall': 0.15698924731182795,
 'f1-score': 0.2517241379310345,
 'support': 465},
'Foreign': {'precision': 0.5957446808510638,
 'recall': 0.08259587020648967,
 'f1-score': 0.14507772020725387,
 'support': 339},
'History': {'precision': 0.62,
 'recall': 0.10954063604240283,
 'f1-score': 0.1861861861861862,
 'support': 283},
'Horror': {'precision': 0.7461368653421634,
 'recall': 0.3595744680851064,
 'f1-score': 0.4852835606604452,
 'support': 940},
'Music': {'precision': 0.6610169491525424,
 'recall': 0.23076923076923078,
 'f1-score': 0.34210526315789475,
 'support': 338},
'Mystery': {'precision': 0.5384615384615384,
 'recall': 0.07763401109057301,
 'f1-score': 0.13570274636510501,
 'support': 541},
```

```
'Romance': {'precision': 0.5656108597285068,
 'recall': 0.18532246108228317,
 'f1-score': 0.2791736460078168,
 'support': 1349},
'Science Fiction': {'precision': 0.8041958041958042,
 'recall': 0.36977491961414793,
 'f1-score': 0.5066079295154184,
 'support': 622},
'TV Movie': {'precision': 0.6666666666666666,
 'recall': 0.050314465408805034,
 'f1-score': 0.09356725146198831,
 'support': 159},
'Thriller': {'precision': 0.6154910096818811,
 'recall': 0.2854393842206543,
 'f1-score': 0.3900087642418931,
 'support': 1559},
'War': {'precision': 0.6875,
 'recall': 0.2661290322580645,
 'f1-score': 0.3837209302325581,
 'support': 248},
'Western': {'precision': 0.8375,
 'recall': 0.29777777777777775,
 'f1-score': 0.43934426229508194,
 'support': 225},
'micro avg': {'precision': 0.6697987948115617,
 'recall': 0.35086405221764483,
 'f1-score': 0.46050136928586477,
 'support': 18691},
'macro avg': {'precision': 0.6833754318479817,
 'recall': 0.25653236528494583,
 'f1-score': 0.357858784000396,
 'support': 18691},
'weighted avg': {'precision': 0.6662539841110997,
 'recall': 0.35086405221764483,
 'f1-score': 0.43934954271511234,
 'support': 18691},
'samples avg': {'precision': 0.5028682170542635,
 'recall': 0.38376603732814735,
 'f1-score': 0.40637185402358406,
 'support': 18691}}
```