Predicting Movie Genres: Presentation

The goal of this project is to build a model that can effectively predict movie genres based on a comprehensive list of features about movies such as movie summary, tagline, budget, revenue, user ratings, users who like the movie, cast and crew members, keywords (to describe movie), etc.

Since movies are not one-dimensional and can span multiple genres, a single label classification model will not suffice. Instead, we need to build a multi label classification model to predict the genres of a movie. The image below illustrates the different types of classification

Table 1

X	y
X_1	t_1
X_2	t_2
X_3	t_1
X_4	t_2
X_5	t_1

Binary Classification

Table 2

X	y
X_1	t_2
X_2	t_3
X_3	t_4
X_4	t_1
X_5	t_3

Multi-class Classification

Table 3

X	y
X_1	$[t_2, t_5]$
X_2	$[t_1, t_2, t_3, t_4]$
X_3	$[t_3]$
X_3	$[t_2, t_4]$
X_3	$[t_1, t_3, t_4]$

Multi-label Classification

models.

Research Question

Can we predict the genre(s) of a movie given its features (such as plot description, ratings, cast and crew members, etc)?

Part I: Collecting the data

We collected comprehensive movie data that is broken up into multiple CSVs from Kaggle. The dataset contained the following CSVs:

- links_small.csv: links to iMDB and tMDB for movies | Not used
- links.csv: links to iMDB and tMDB for fewer movies | Not used
- ratings_small.csv: user rating for fewer movies | Not used

- ratings.csv: user rating for many movies | Used
- credits.csv: crew and cast information about each movie | Used
- keywords.csv: keywords for each movie | Used
- movies_metadata.csv: information about each movie | Used

```
import pandas as pd
from google.colab import drive
from IPython.display import Image
drive.mount('drive')
data_dir = "drive/My Drive/301 Project/Data/"
image_dir = "drive/My Drive/301 Project/Data/images/"
```

```
pd.read_csv(data_dir + "ratings.csv").head(1)
```

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529

```
pd.read_csv(data_dir + "credits.csv").head(1)
```

	cast	crew	id
0	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...	862

```
pd.read_csv(data_dir + "keywords.csv").head(1)
```

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...

```
pd.read_csv(data_dir + "movies_metadata.csv", low_memory=False).head(1)
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	T

Part II: Cleaning the data

```
pd.read_csv(data_dir + "ratings.csv")
df_movies = pd.read_csv(data_dir + "cleaned_dataframe.csv").drop(
    columns=["Unnamed: 0"],
    axis=1
)
cols = df_movies.columns
df_movies.head(1)
```

	Animation	Horror	Mystery	Fantasy	Romance	TV Movie	Family	Documentary	Western	Science Fiction	Foreign	Crime	Adventur
0	1	0	0	0	0	0	1	0	0	0	0	0	

We cleaned the each CSV in the following manner:

- ratings.csv:
  - We used the `userId` and the `rating` column to determine which userIds liked each movie and stored this in `userId_who_like`
  - We took the average rating from all users for a movie and stored this in a column called `rating_average`

```
df_movies[cols[37:]].head(1)
```

	rating_average	userId_who_like
0	3.59893	2103 6177 6525 8659 9328 9682 11214 13839 1523...

- credits.csv:
  - We extracted `Screenplay`, `Director`, and `Composer` from the `crew` column
  - We transformed `cast` into a string of cast member names with each name underscored (instead of spaces) and all names space seperated.

```
df_movies[cols[32:36]].head(2)
```

	Composer	Director	Screenplay	cast
0	NaN	John_Lasseter	Alec_Sokolow	Tom_Hanks Tim_Allen Don_Rickles Jim_Varney Wal...
1	James_Horner	Joe_Johnston	Jim_Strain	Robin_Williams Jonathan_Hyde

- keywords.csv:
  - We transformed `keywords` into a string of keywords with each keyword underscored (for spaces) and all keywords apce seperated

```
df_movies[cols[36:37]].head(2)
```

	keywords
0	jealousy toy boy friendship friends rivalry bo...
1	board_game disappearance based_on_children's_b...

- movies_metadata.csv:

- We only preserved some columns from this dataset: `budget, id, original_title, overview, popularity, production_companies, revenue, runtime, tagline, vote_average, vote_count, [genres]`
- We transformed `collection` into the name of the collection with underscore replacing space
- We transformed `production_companies` into a string of companies with each company underscored and all companies space seperated.
- We also added a column for each genre as it would be useful for visual exploration and building a machine learning model. If a movie belonged to a certain genre, the value of the movie for that that genre would be 1, else 0.

```
df_movies[cols[:32]].head(2)
```

	Animation	Horror	Mystery	Fantasy	Romance	TV Movie	Family	Documentary	Western	Science Fiction	Foreign	Crime	Adventur
0	1	0	0	0	0	0	1	0	0	0	0	0	
1	0	0	0	1	0	0	1	0	0	0	0	0	

We merged these datasets to produce a complete movies dataset with 39 columns, 20 of which were for genres.

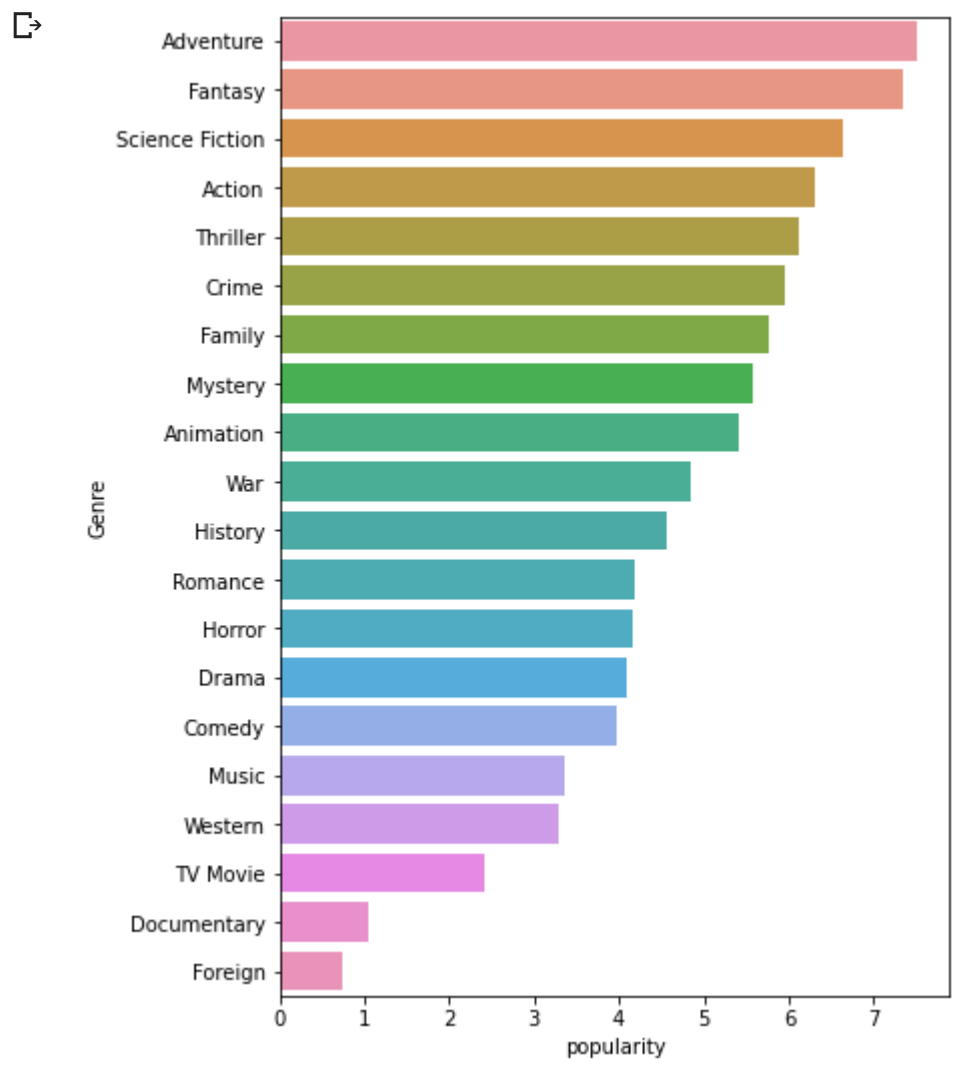
Part III: Exploring the data

In this section, we explored the relationships between genres and different categorical and quantitative features in our movies dataset by using bar plots and word clouds.

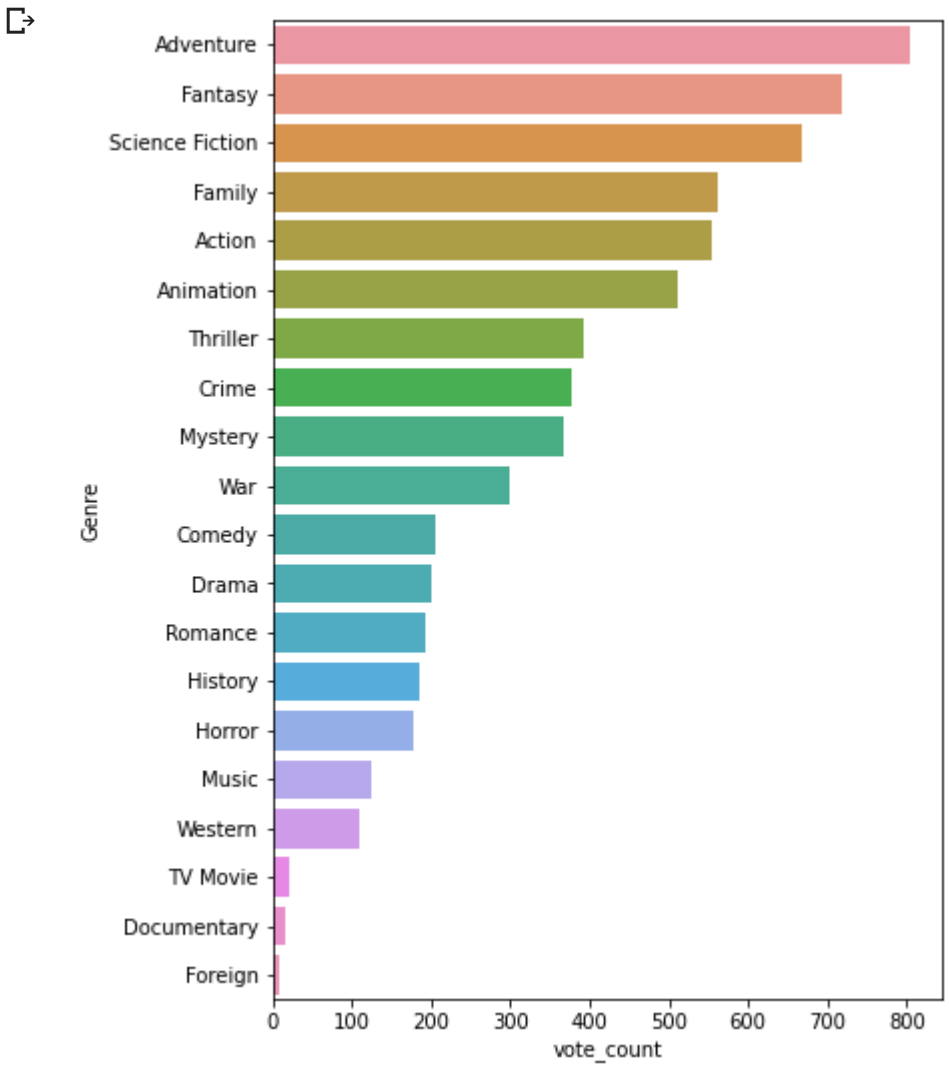
Quantitative features useful in determining genres

In the following bar plots, there is a clear difference between different genres for the feature whose average is being plotted.

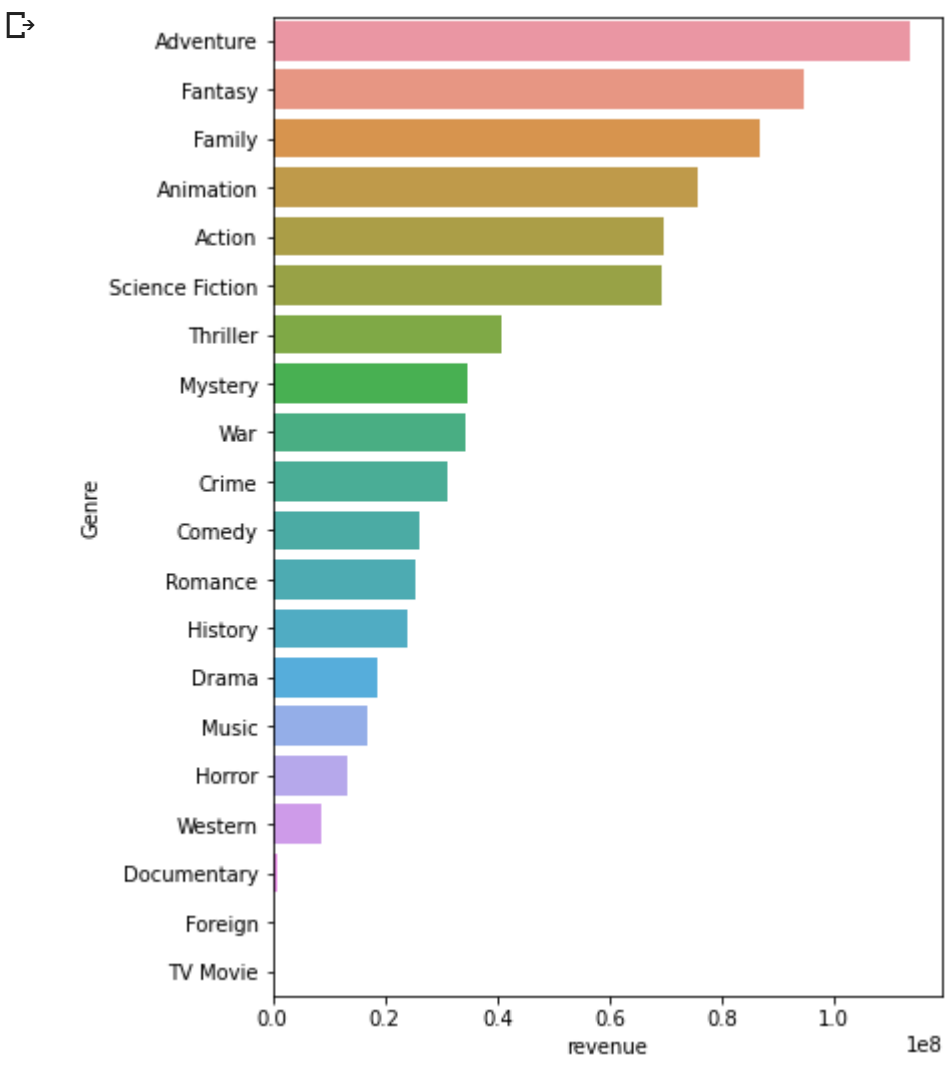
```
Image(image_dir + "Average_popularity_plot.png")
```




```
Image(image_dir + "Average_vote_count_plot.png")
```

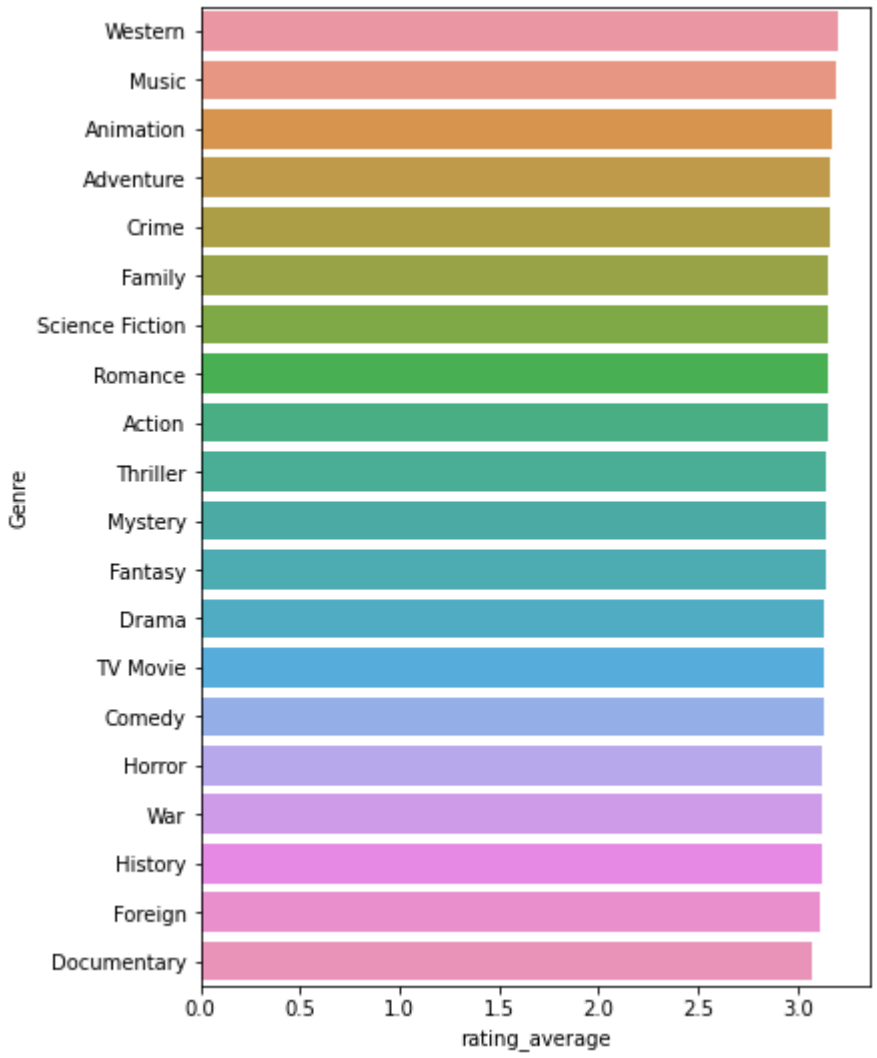



```
Image(image_dir + "Average_revenue_plot.png")
```



Quantitative features not as useful in determining genres

```
Image(image_dir + "Average_rating_average_plot.png")
```

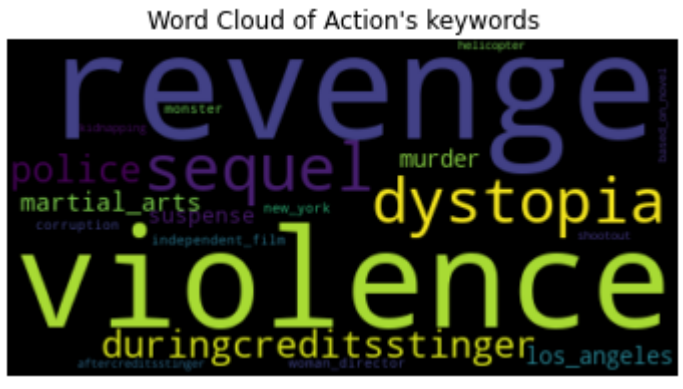


Categorical features useful in determining genres

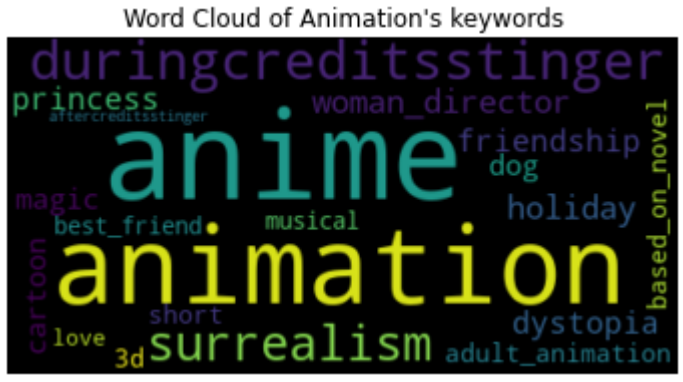
We generated word clouds for all genres for various categorical features. The keywords and cast members associated with distict genres were accurately depicted in the words clouds (examples shown below) and thus these features are worth considering.

Keywords

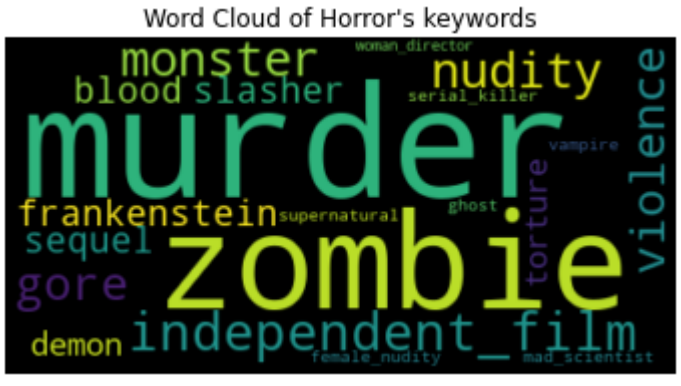
```
Image(image_dir + "Action_keywords.png")
```



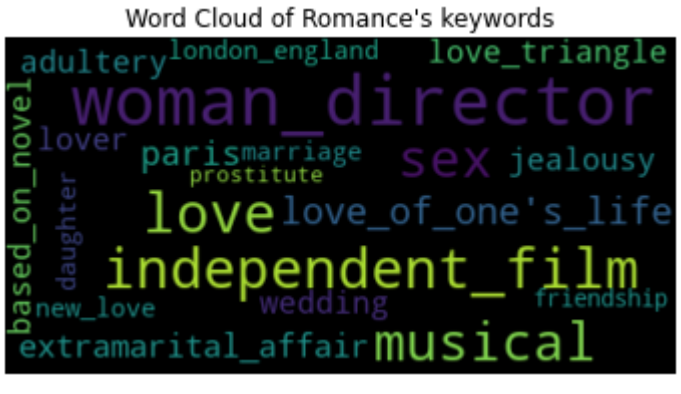

```
Image(image_dir + "Animation_keywords.png")
```




```
Image(image_dir + "Horror_keywords.png")
```

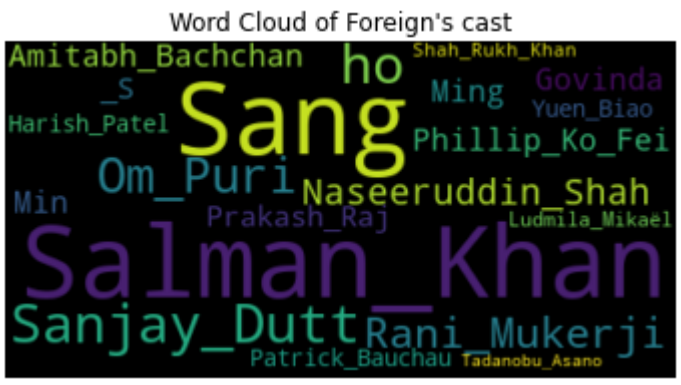



```
Image(image_dir + "Romance_keywords.png")
```

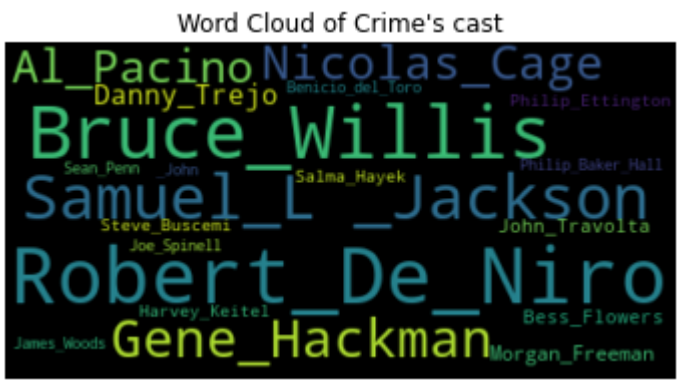


Cast

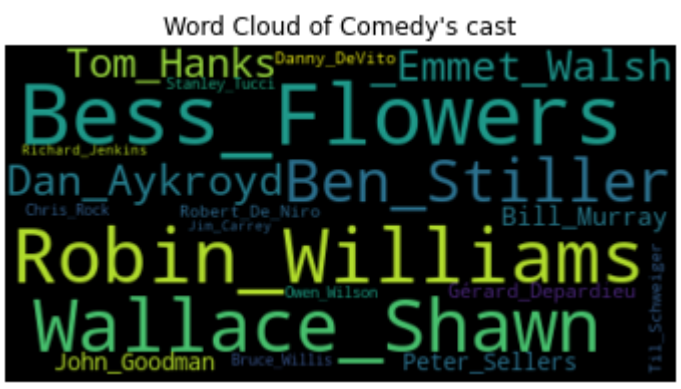
```
Image(image_dir + "Foreign_cast.png")
```




```
Image(image_dir + "Crime_cast.png")
```

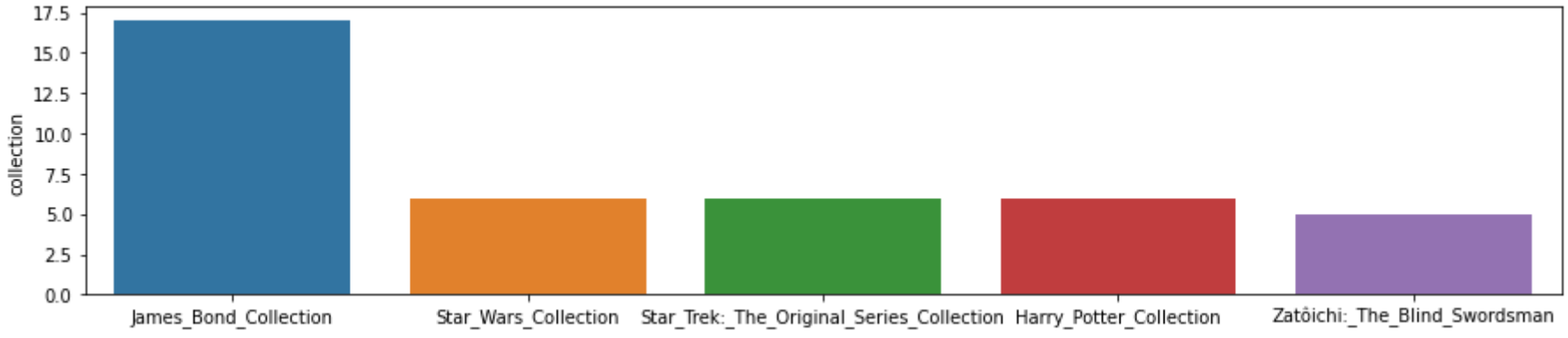



```
Image(image_dir + "Comedy_cast.png")
```



We also plotted bar charts for the most frequent `production_companies`, `collections`, `Composer`, `Director`, and `Screenplay` for a few genres and did light research to determine which features were worth considering. Only `collection` produced reasonable results:

```
Image(image_dir + "Adventure_collection.png")
```



Part IV: Building an ML model