# Understanding Film Rating Data

**Final Project Report**
**W200 - Python Fundamentals for Data Science, Summer 2018, Section 1**

**Team members: Eduarda Espindola, Payman Roghani, Mark Yong**

## Context:

With the abundance of channels available to watch movies and the increasing number of movie titles, it is challenging and time-consuming for consumers to find the content that match their preferences and tastes. In addition to traditional channels like TV networks and movie theaters, there are now other online platforms with original content as well as curated lists of movies not available on other platforms. As a result, there is reliable recommendation systems are needed to provide suggestions to consumers to encourage more usage and increase satisfaction. Some platforms (e.g. Netflix) provide recommendations based on users' ratings applied to films or shows they watch, as well as the content they have previously watched, but the results are not always satisfactory.

## Objective:

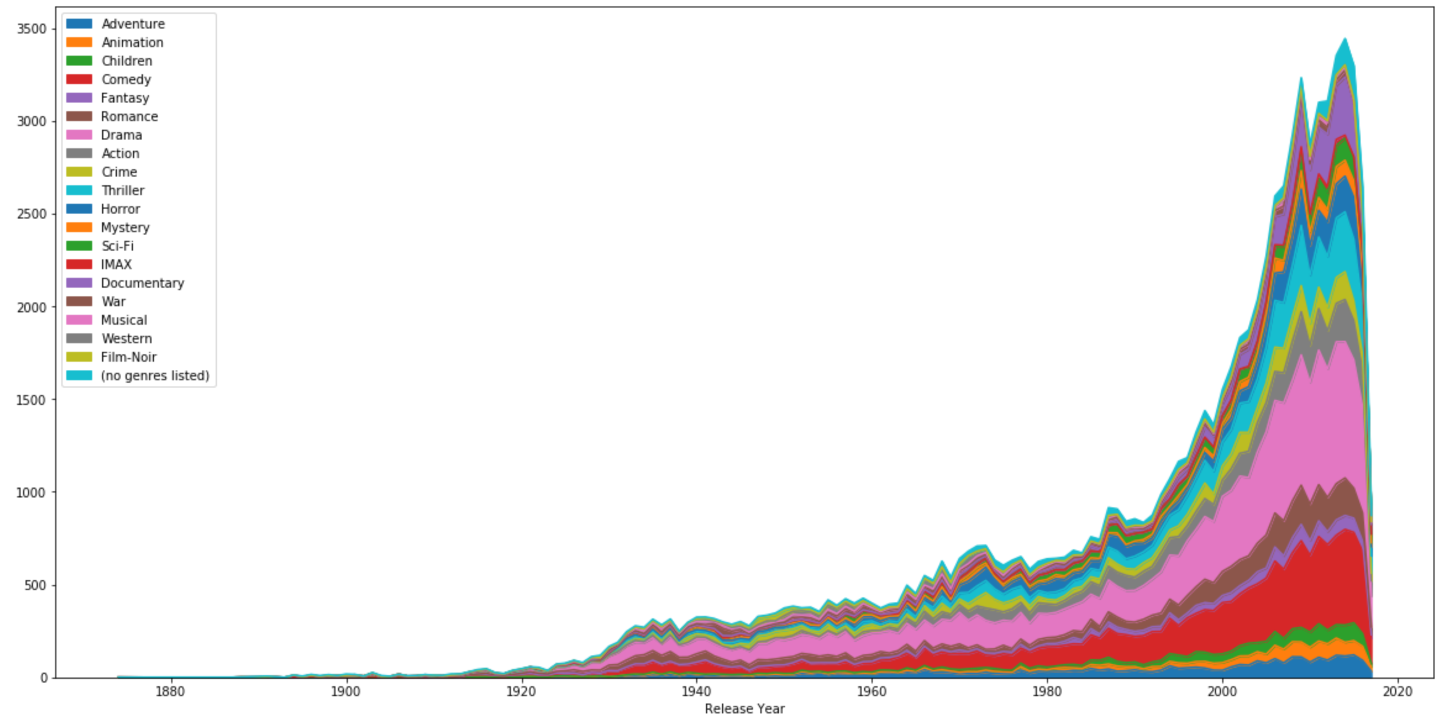
As stated in our proposal, the main purpose of this project is to better understand how users apply ratings and tags to movies, and find potential correlations between ratings and other parameters. That said, upon further analysis of our data, we realized that for a deep understanding of user ratings, we would need to do statistical analysis (e.g. cluster analysis, regression), which would be out of the scope of this course. Therefore, we focused our efforts and analysis on factors and parameters that could potentially impact movie ratings and revenue (i.e. popularity).

## Datasets:

- MovieLens
    - MovieLens 20M Dataset (includes 20 million ratings and 465,000 tags applied to 27,000 movies by 138,000 users)
    - Link: https://grouplens.org/datasets/movielens/20m/
    - Key columns in the dataset:
        - movie title, release year, genre, ratings (user-generated ratings, made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars), user id (anonymized), tmdb id (to link to TMDB data)

| File | Columns | Number of Rows |
|---|---|---|
| movies.csv | movie id, title (release year), [genres, '\|' separated] | 27,278 |
| links.csv | movie id, tmdb id, imdb id | 27,009 |
| ratings.csv | user id, movie id, rating, timestamp | 20,000,263 |

- TMDB (The Movie Database):
    - Key data for our project: ratings, cast and crew, plot keywords, similar movies, reviews
    - Link to API description: [https://www.themoviedb.org/documentation/api?language=en](https://www.themoviedb.org/documentation/api?language=en)
    - Key columns in the dataset:
        - tmdb id, vote average (scale of 1-10), popularity (ad index based on number of page views, number of user ratings/watchlist/favourite additions) cast/crew, budget, revenue, release date (YYYY-MM-DD)
- Initial Plots:

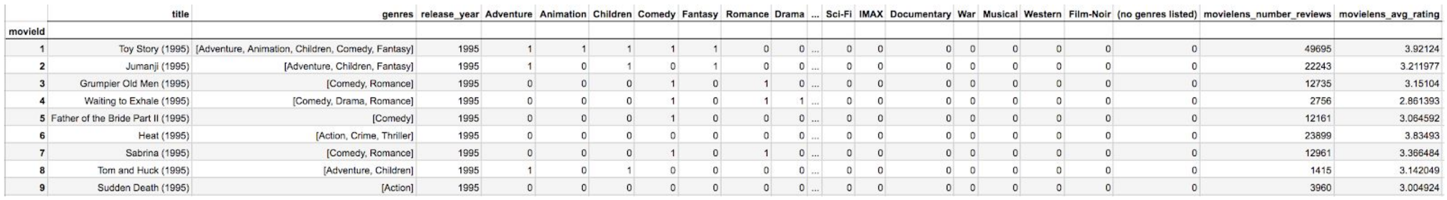


**Data Exploration and Reconciliation:**

- After loading all three databases from MovieLens (movies, links, ratings), we wanted to get them in one unique major dataset. For the movies table, we had:
    - movieId - unique identifier for the movie in the MovieLens Database
    - Title - the title for the movie, followed by the release year inside parenthesis
    - Genres - each genre that the movie fits into, separated by pipes.
- The year might be a relevant variable in further analysis, so we have created a release_year column, that gets its values from splitting by "(" and ")" in the title column to get the release year of the movie.
- To enable a deeper genre analysis, we have created, for each movie, a list of genres, separating the string by pipes. Then, for all the movies, we have generated a list with all unique genre values. For each genre, in the main movies table, we created a column, that will receive the value 1 if the movie fits into that genre and 0 if it doesn't.

- In the ratings table, each movie (movieId) has several and a varying number of reviews and grades. Each rating has the information: movieId (MovieLens unique identifier for the movie), userId (MovieLens unique identifier for the person that made a review, rating (the actual rating given for the movie, ranging from 0 to 5 in 0.5 intervals) and timestamp (date and time of the submission of the review.
- In the links table, we have the movieId (MovieLens unique identifier for the movie) and their corresponding tmdbid (TMDB unique identifier for the movie) and imdbid (IMDB unique identifier for the movie)
- We grouped the ratings table by each movie - movieId  (counting the number of reviews and taking the average of the ratings as aggregation methods). Then we did a left join of the movies table with the grouped ratings table, and we were able to add, to the movies table, a column that accounted the number of reviews per movie (movielens_number_reviews) and the average rating given to each movie (movielens_avg_rating).

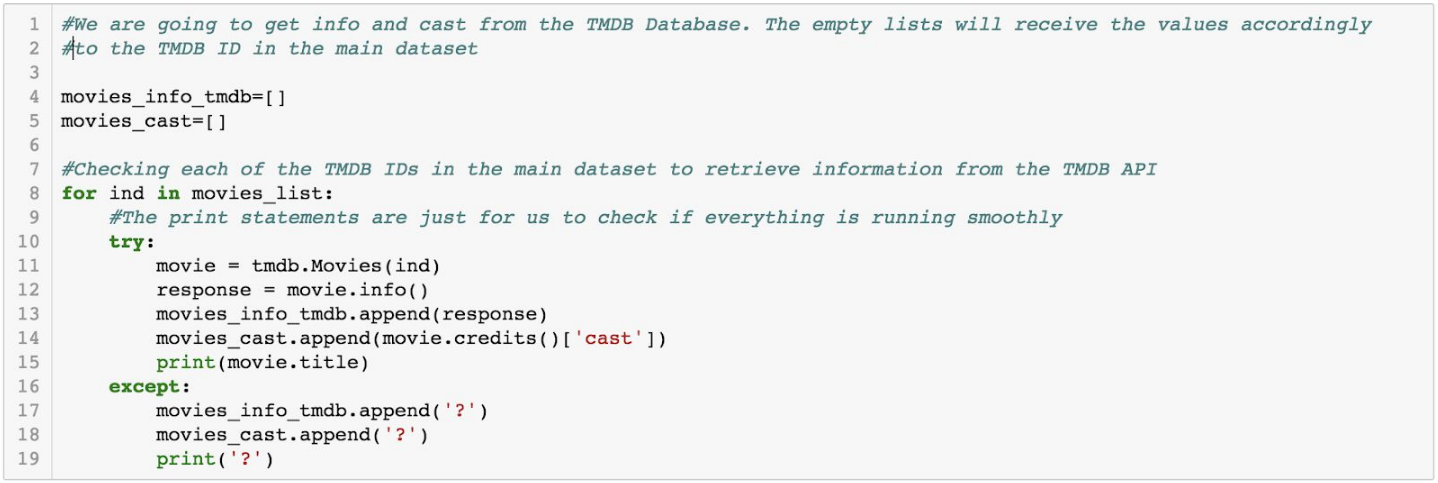
After that we had this table:



| | title | genres | release_year | Adventure | Animation | Children | Comedy | Fantasy | Romance | Drama | ... | Sci-Fi | IMAX | Documentary | War | Musical | Western | Film-Noir | (no genres listed) | movielens_number_reviews | movielens_avg_rating |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| movieId | | | | | | | | | | | | | | | | | | | | | |
| 1 | Toy Story (1995) | [Adventure, Animation, Children, Comedy, Fantasy] | 1995 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 49695 | 3.92124 |
| 2 | Jumanji (1995) | [Adventure, Children, Fantasy] | 1995 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22243 | 3.211977 |
| 3 | Grumpier Old Men (1995) | [Comedy, Romance] | 1995 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12735 | 3.15104 |
| 4 | Waiting to Exhale (1995) | [Comedy, Drama, Romance] | 1995 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2756 | 2.861393 |
| 5 | Father of the Bride Part II (1995) | [Comedy] | 1995 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12161 | 3.064592 |
| 6 | Heat (1995) | [Action, Crime, Thriller] | 1995 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23899 | 3.83493 |
| 7 | Sabrina (1995) | [Comedy, Romance] | 1995 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12961 | 3.366484 |
| 8 | Tom and Huck (1995) | [Adventure, Children] | 1995 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1415 | 3.142049 |
| 9 | Sudden Death (1995) | [Action] | 1995 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3960 | 3.004924 |

However, there were other information that we needed, like casting, revenues, etc, that were unavailable in the movielens database. At that point we knew we wanted to focus on movies released in the last ten years, so we set as our new main table a subset of the previous one, which relase_years is greater than or equal to 2008. Finally, we resorted to the links table, and used an inner join to get, for each movieId (MovieLens ID) the corresponding tmdb id, and stored it in our main movies database.

Having the corresponding the tmdbid for each movie enabled us to gather data from the TMDB database, via the API **tmdbsimple**:



```python
#We are going to get info and cast from the TMDB Database. The empty lists will receive the values accordingly
#to the TMDB ID in the main dataset

movies_info_tmdb=[]
movies_cast=[]

#Checking each of the TMDB IDs in the main dataset to retrieve information from the TMDB API
for ind in movies_list:
    #The print statements are just for us to check if everything is running smoothly
    try:
        movie = tmdb.Movies(ind)
        response = movie.info()
        movies_info_tmdb.append(response)
        movies_cast.append(movie.credits()['cast'])
        print(movie.title)
    except:
        movies_info_tmdb.append('?')
        movies_cast.append('?')
        print('?')
```

Later, it was only a matter of creating two new columns (tmdb_info and tmdb_cast) and storing the values on them. For some movies, it was not possible to get any information in the tmdb database. Those movies where dropped off the analysis. The final generated database was then uploaded to a csv to enable other manipulations without having to get each value from the TMDB API. However, when loading the csv, the dictionary in the tmdb_info column and the dictionary in the cast column came as strings, and for us to get the other variables we intended to get from the tmdb database, some string manipulation and splitting was necessary:

```python
#Just fixing the tmdb_cast column so we can better use the data -  it comes as a string,
#but it should be a dictionary, so we will just use split to get what we want in the form of a list

cast_all=[]
cast_main=[]
for i in range(len(movies)):
    cast_str=movies.iloc[i]["tmdb_cast"]
    cast_spl=cast_str.split("'name': ")
    actors_list=[]
    for j in range(1,len(cast_spl)):
        actor_name=cast_spl[j].split(',')[0]
        actor=actor_name.replace("'","").replace('"','')
        actors_list.append(actor)
    cast_all.append(actors_list)
    #The first three actors that appear are the main ones
    cast_main.append(actors_list[0:3])
```

```python
#Creating two new columns for our main dataset, to better access cast members and main actors
movies["actors_list"]=cast_all
movies["main_actors"]=cast_main
```

```python
#Using a similar split approach to better process the information in the tmdb_info column

production_companies=[]
release_dates=[]
votes_average=[]
votes_count=[]
revenues=[]
budgets=[]
for i in range(len(movies)):
    info_str=movies.iloc[i]["tmdb_info"]
    budget=int(info_str.split("'budget': ")[1].split(",")[0].replace("'","").replace('"',''))
    vote_average=float(info_str.split("'vote_average': ")[1].split(",")[0].replace("'","").replace('"',''))
    vote_count=int(info_str.split("'vote_count': ")[1].split("}")[0].replace("'","").replace('"',''))
    release_date=info_str.split("'release_date': ")[1].split(",")[0].replace("'","").replace('"','')
    revenue=int(info_str.split("'revenue': ")[1].split(",")[0].replace("'","").replace('"',''))
    production_spl=info_str.split("'production_companies': ")[1].split("}]")[0].split("'name': ")
    production=[]
    for j in range(1,len(production_spl)):
        company_name=production_spl[j].split(',')[0]
        company=company_name.replace("'","").replace('"','')
        production.append(company)
    cast_all.append(actors_list)
    production_companies.append(production)
    release_dates.append(release_date)
    votes_average.append(vote_average)
    votes_count.append(vote_count)
    revenues.append(revenue)
    budgets.append(budget)
```

```python
#Creating the columns for better processing the information on the tmdb_info in the movies database

movies['production_companies']=production_companies
movies['release_date']=release_dates
movies['tmdb_vote_avg']=votes_average
movies['vote_count']=votes_count
movies['revenue']=revenues
movies['budget']=budgets
movies.head(10)
```

Finally, we get our main final dataset:

[illegible]

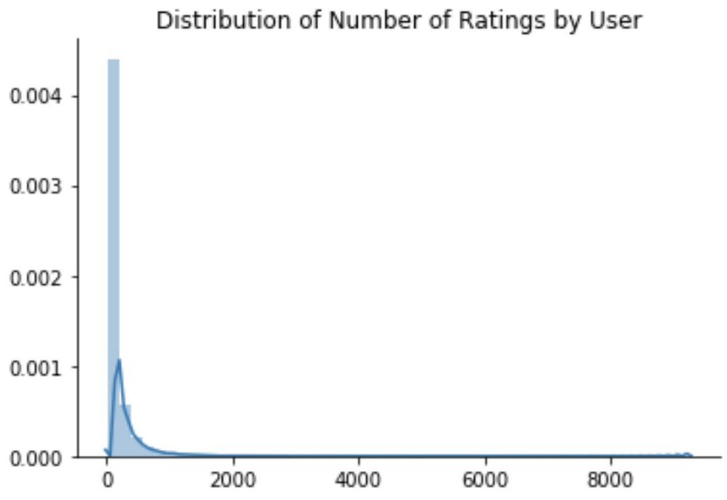
**P_L is only revenue minus budget*

**General Insights and Statistics:**

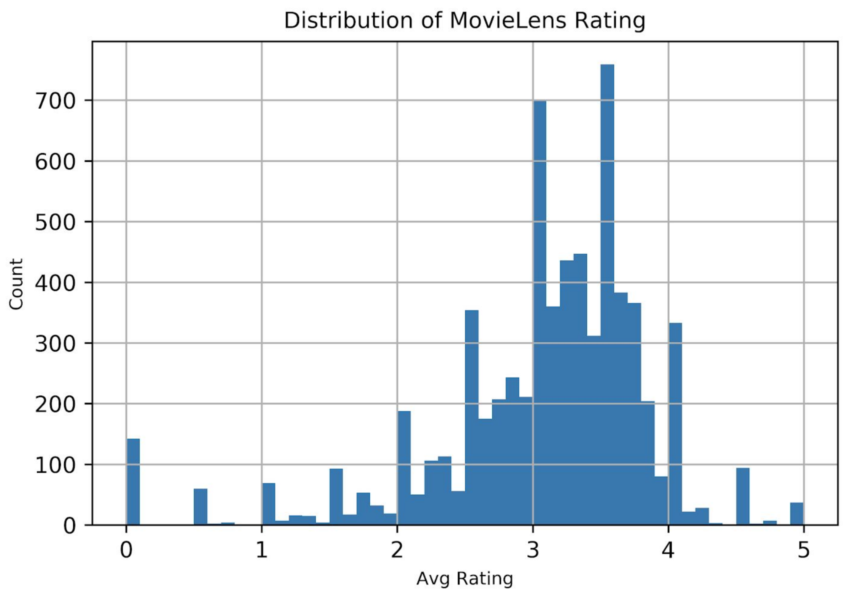
- ➢ Top 10 movies based on MovieLens rating
  - ○ Top 10 movies are mostly from action and adventure genres, not surprisingly

| Title | Genres | Release Year | Budget ($MM) | Revenue ($MM) | MovieLens Rating | Popularity | TMDB Rating |
|---|---|---|---|---|---|---|---|
| The Dark Knight | ['Action', 'Crime', 'Drama', 'IMAX'] | 2008 | 185 | 1,005 | 4.2 | 28.9 | 8.4 |
| Inception | ['Action', 'Crime', 'Drama', 'Mystery', 'Sci-Fi', 'Thriller', 'IMAX'] | 2010 | 160 | 826 | 4.2 | 32.8 | 8.2 |
| WALL¬E | ['Adventure', 'Animation', 'Children', 'Romance', 'Sci-Fi'] | 2008 | 180 | 521 | 4.0 | 29.3 | 7.9 |
| Up | ['Adventure', 'Animation', 'Children', 'Drama'] | 2009 | 175 | 735 | 4.0 | 22.3 | 7.8 |
| Django Unchained | ['Action', 'Drama', 'Western'] | 2012 | 100 | 425 | 4.0 | 18.1 | 7.9 |
| The King's Speech | ['Drama'] | 2010 | 15 | 414 | 4.0 | 14.7 | 7.7 |
| Moon | ['Drama', 'Mystery', 'Sci-Fi', 'Thriller'] | 2009 | 5 | 10 | 4.0 | 12.2 | 7.6 |
| Inglourious Basterds | ['Action', 'Drama', 'War'] | 2009 | 70 | 319 | 4.0 | 15.1 | 8.0 |
| Toy Story 3 | ['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy', 'IMAX'] | 2010 | 200 | 1,067 | 4.0 | 17.2 | 7.7 |
| How to Train Your Dragon | ['Adventure', 'Animation', 'Children', 'Fantasy', 'IMAX'] | 2010 | 165 | 495 | 4.0 | 19.0 | 7.6 |

- ➢ Number of rating by user
  - ○ While there are users on MovieLens that are highly engaged and have provided ratings to thousands of movies, the average rating per user is 144



- ➢ Number of rating by user
  - ○ Most ratings on MovieLens fall between 3 to 4 out of 5. There are rarely any movies with rating below 1 above 4. That said, there are about 150 movies with 0 rating and about 150 movies with 4.5 or 5 ratings.



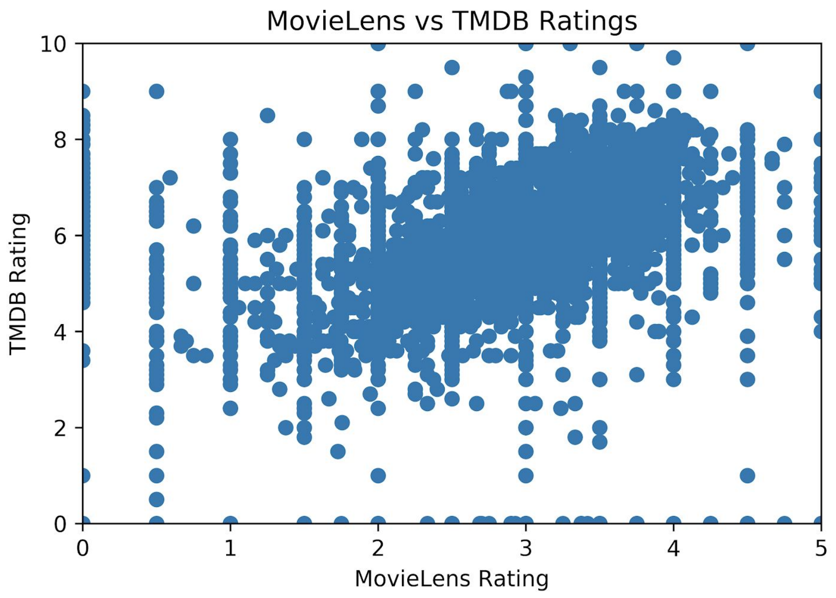
**Our Approach:**

- Decided to focus on movies released 2008-2017
- While we initially planned to use IMDB dataset as well, we decided not to because the other 2 sources had sufficient data for our analysis
- Since statistical analysis for user and movie clustering would be out of the scope of this course, we focused our effort on analyzing correlations among rating, popularity (an index included in TMDB dataset and revenue (as a proxy for popularity)

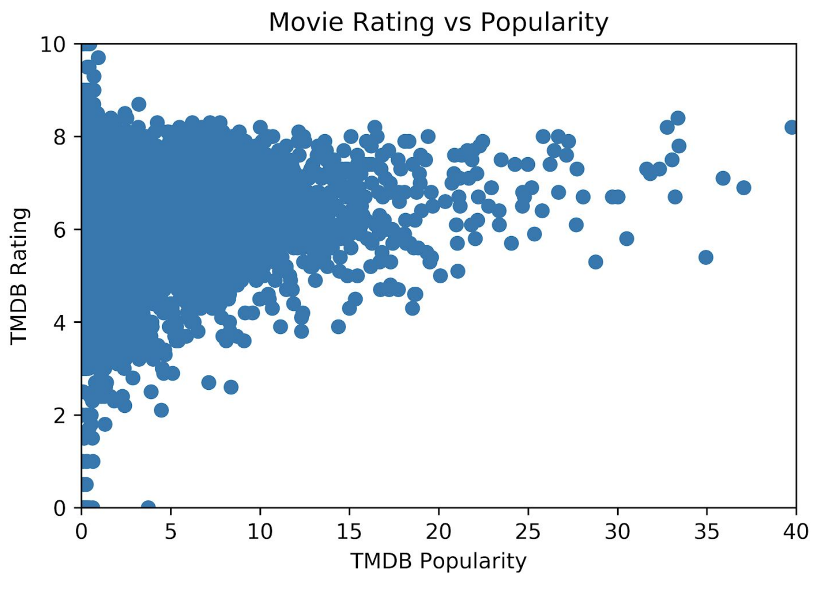
**Key Insights:**

➢ Are movie ratings similar across platforms?

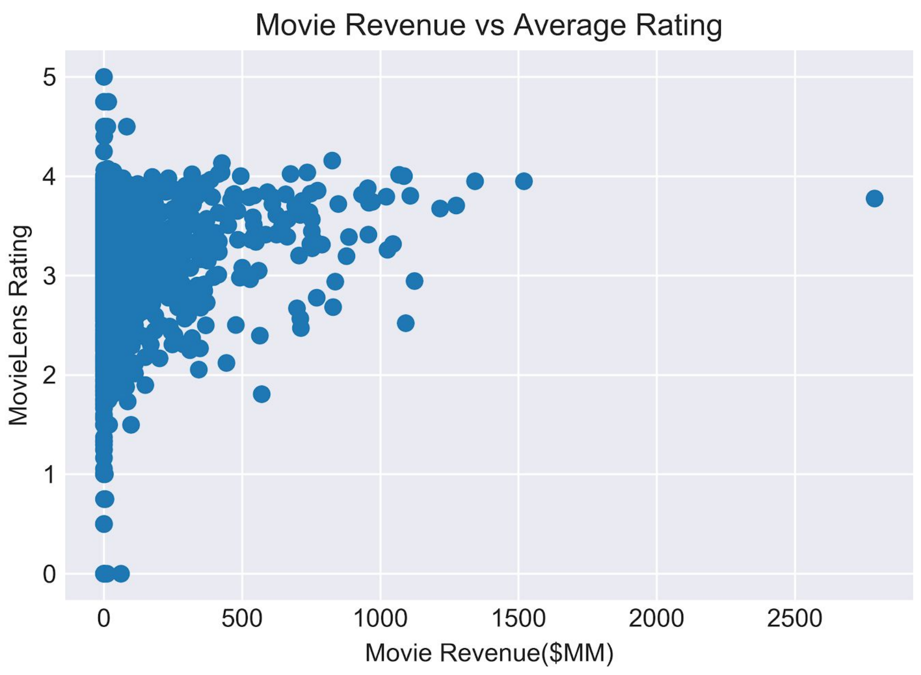
- As seen in the chart below, it seems that while user ratings are not exactly the same on MovieLens and TMDB, there is a close correlation between the two. Therefore we only used MovieLens ratings in our further analysis.



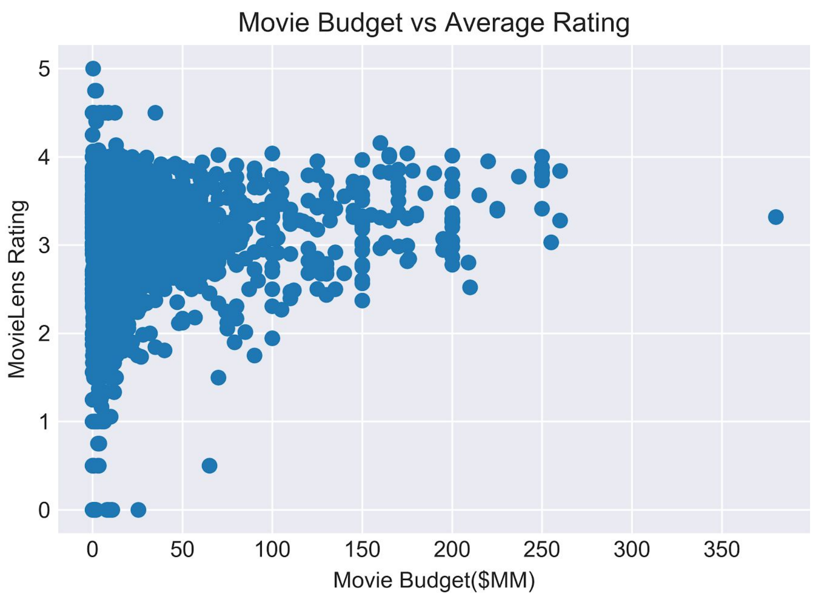
- Is movie popularity correlated with rating?
  - While we did not find a close correlation between popularity index on TMDB and user ratings on that platform, it seem that popular movies have above average rating.



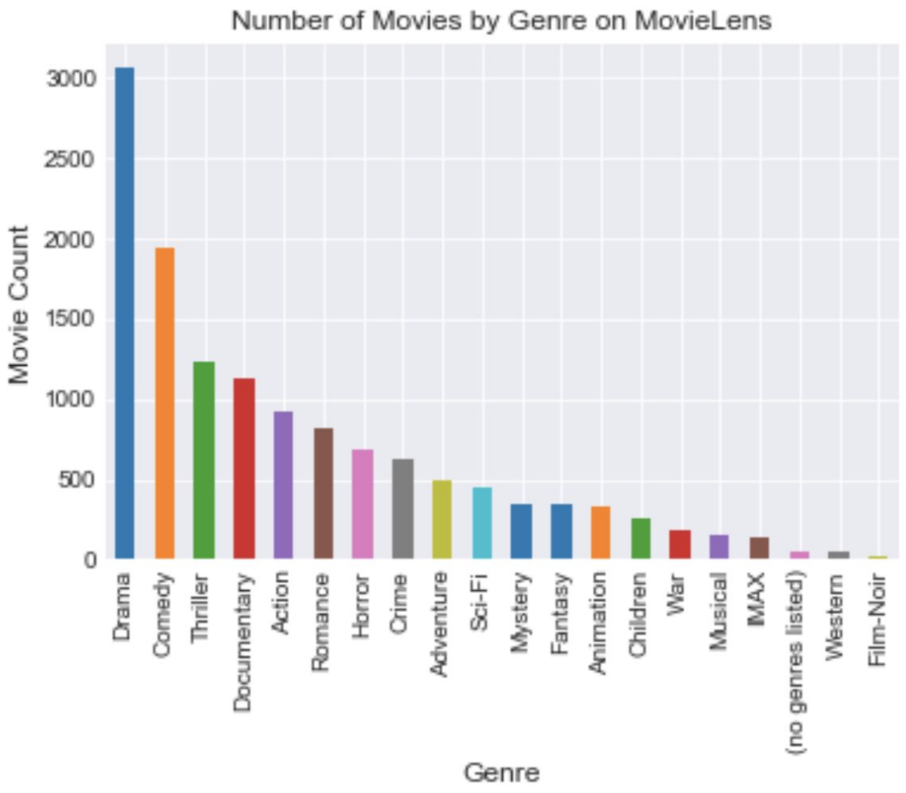
- Is movie revenue (i.e. popularity) a proxy for user-generated rating?
  - The correlation between average user rating on MovieLens and movie revenue is not very close. However, in the chart below we can see a trend, where movies with high revenue have higher than average rating.



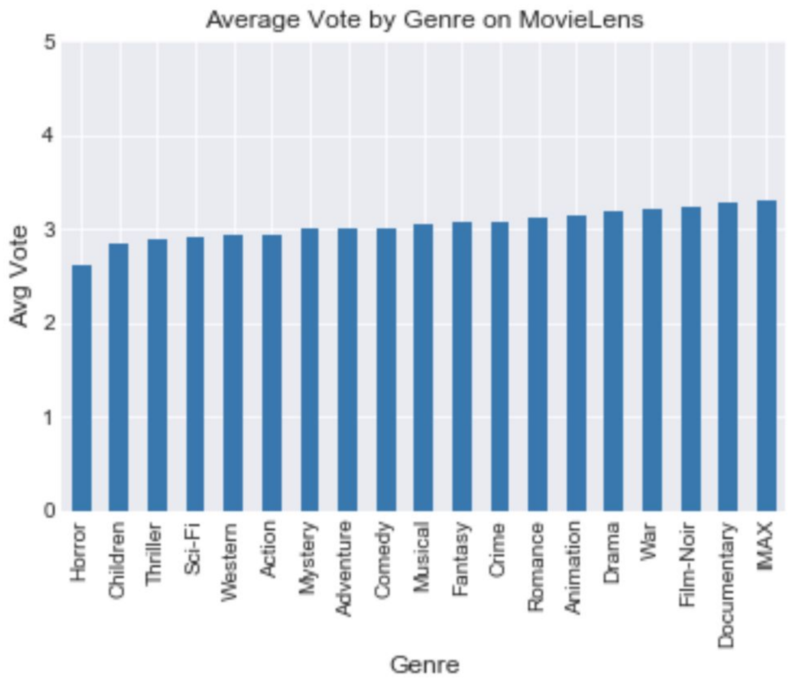
- ➢ Do high-budget movies get higher ratings?
    - ○ Similar to the trend in movies with high revenue, as seen in the chart above, it seems that movies with higher budget tend to get higher user rating on MovieLens.



- ➢ Are specific film genres correlated with higher ratings?
    - ○ Drama genre has the highest frequency among other genres in datasets we analyzed. Comedy and thriller come second and third respectively.

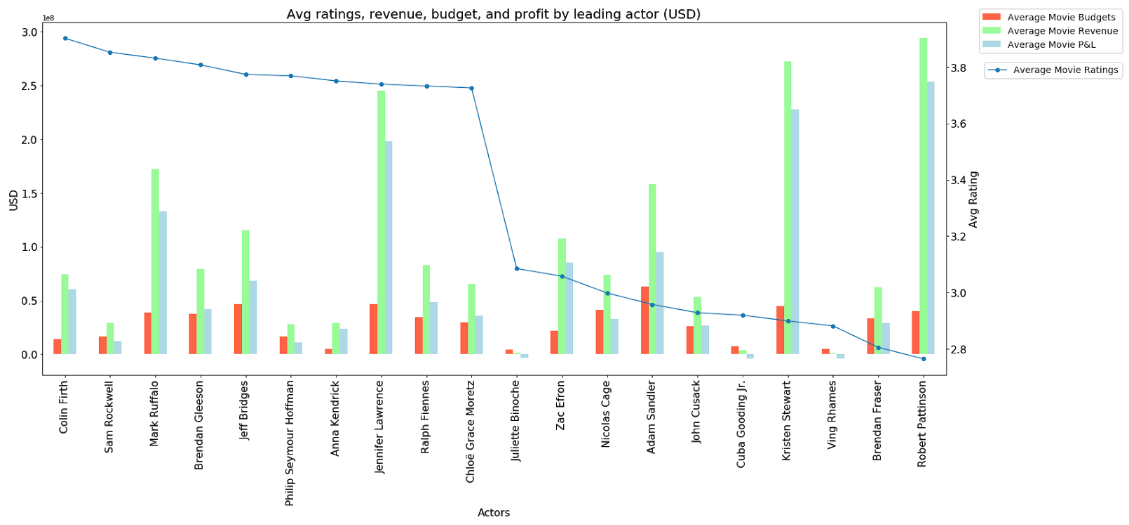


- Based on our analysis, some genres tend to get higher user rating on MovieLens, specifically IMAX (not a real film genre), documentary and film-noir have the highest ratings among other genres. One important caveat is that a lot movies in the dataset have more than 1 genre. The popular movie “The Dark Knight” has 4 genres assigned to it: action, crime, drama and IMAX.



➢ What’s the impact of movie cast on ratings and revenue?

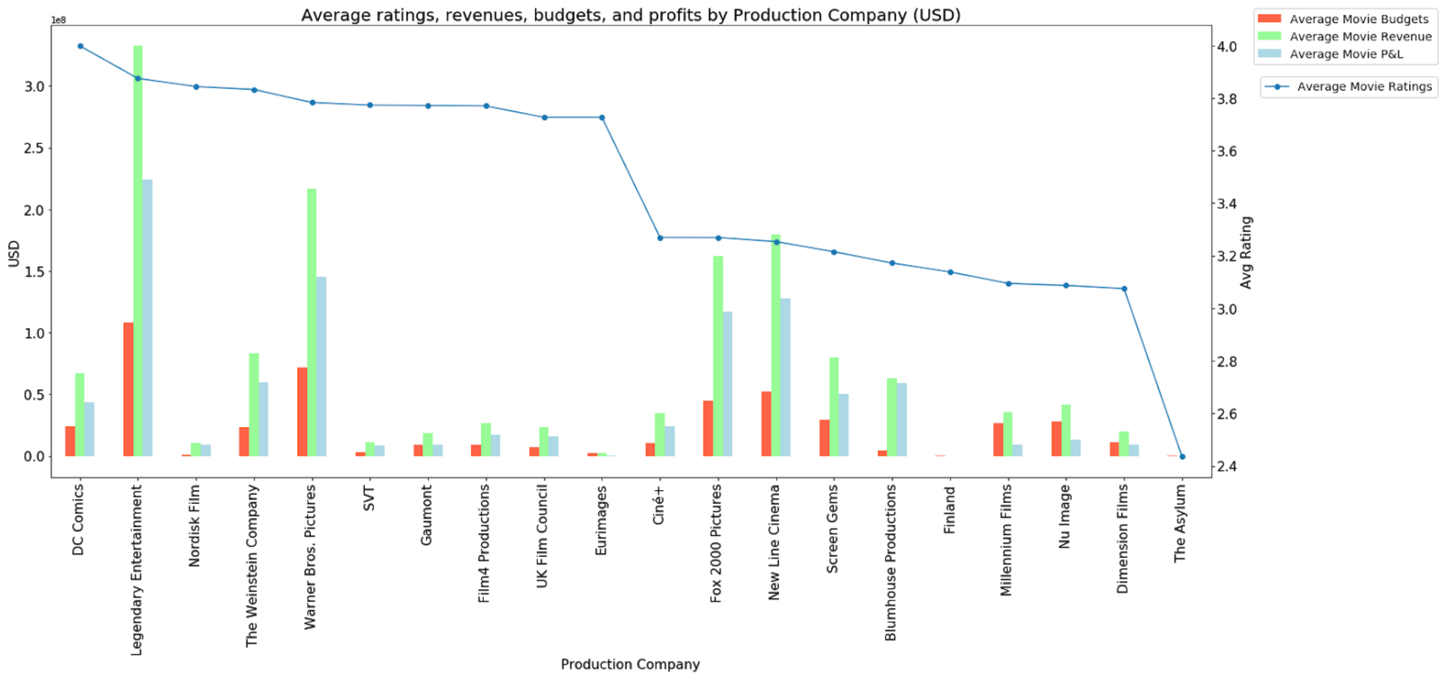
Here we analyse average budget, revenue, P&L and rating of movies starred in the last ten years by each actor. To try to avoid the effects of actors that starred few movie, we have filtered only actors that have starred in over 10 movies in this period. To facilitate the analysis and the visualization, we are only showing top and bottom 10 actors when it comes to movies average rating.



By analysing this chart we can see that, at least for casting, high ratings don’t necessarily mean high revenues. Colin Firth, for instance, for actors that starred over 10 movies in the past 10 years, has the highest average movie rating, while Robert Pattinson has the lowest, and yet, movies with Robert Pattinson as one of the lead actors have generated a much higher revenue, in average, than movies starred by Colin Firth.

➢ What's the impact of Production Company in movie ratings and revenues?

Here we analyse average budget, revenue, P&L and rating of movies produced in the last ten years by each company. To try to avoid the effects of companies that only produced a little number of movies, we have filtered only companies that have produced over 20 movies in this period. To facilitate the analysis and the visualization, we are only showing top and bottom 10 companies when it comes to movies average rating.



Unlikely the actor analysis, at least for production companies, high rated movies seem to translate into higher revenues (compare major US Production Companies: Legendary, Weinstein, Warner Brothers, Fox and New Line - higher average ratings are usually combined with higher movie revenues).

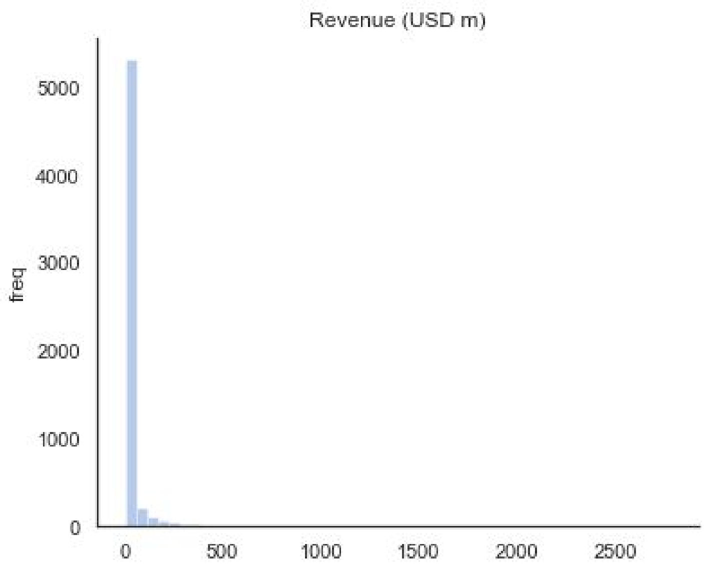
The actor and production company analysis indicate to us that some actors might serve as a bait to get people to watch the movie, and thus generate high revenue, but people won't necessarily like the movie: they want to see that specific actor, but that is not enough for them to rate the movie high. For production companies however, the revenues grow bigger with the ratings, indicating that producing a good movie generates the person-to-person buzz that will drive revenues up.

**Revenue, Budget, Profit**

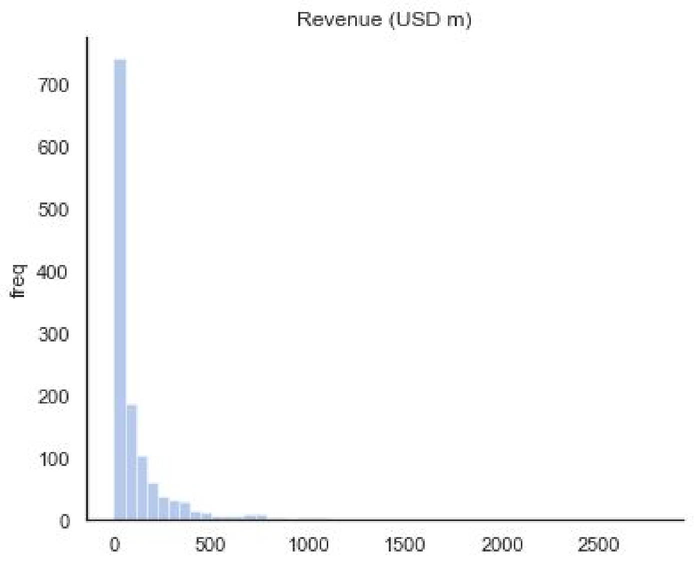
We planned to use revenue and profit as proxies for the popularity of the data, and budget as a predictor.

When first examining the data, revenue and budget appeared to have a large number of zero values. Upon cross-examining the data against Wikipedia and other sources, we found that many of the movies that supposedly had zero revenues or budgets had actual figures, e.g. The Uninvited, which made 41.6 million USD in revenue. Given that these metrics appeared to contain missing values, we decided to drop these rows when graphing or performing comparisons in order to prevent drawing the wrong conclusions.

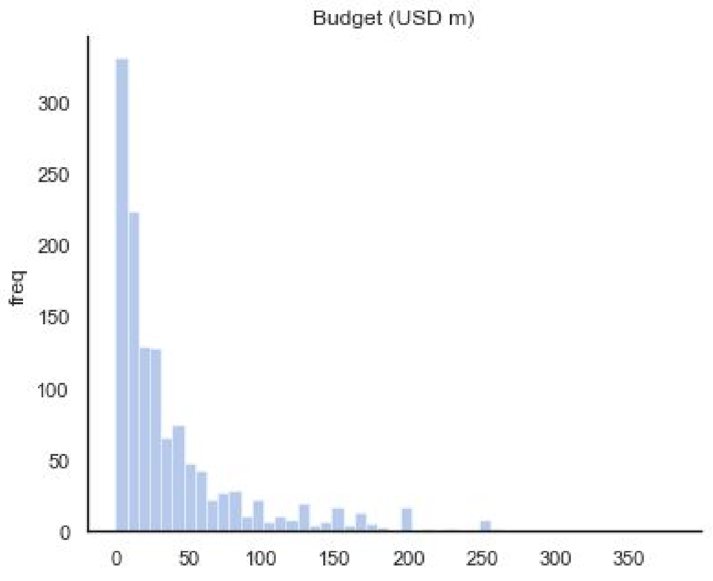
Revenue before correction

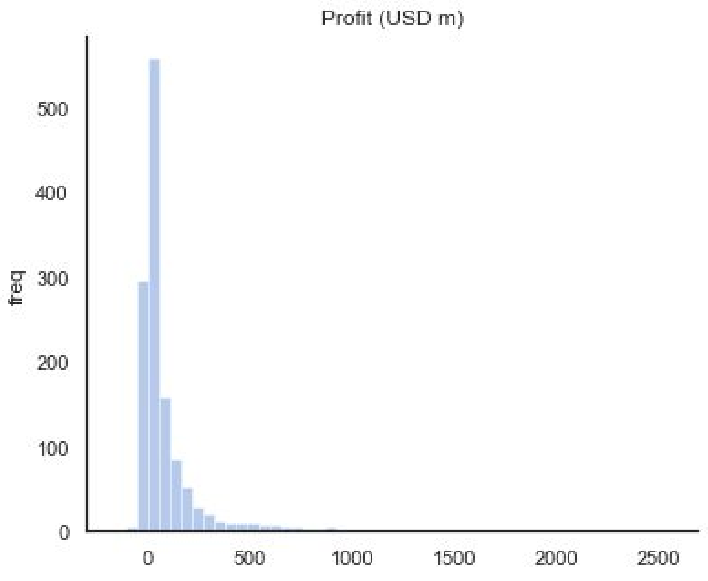


Revenue after correction



Exploring budget and profit, we see the same positive skew

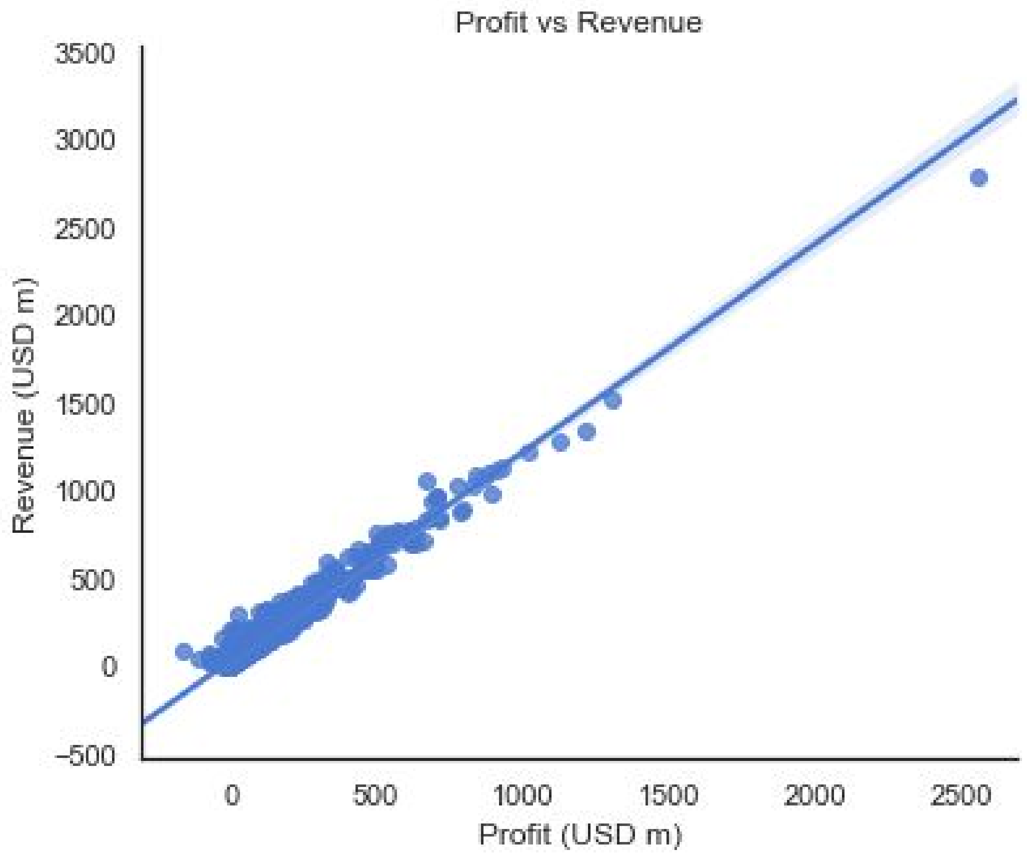




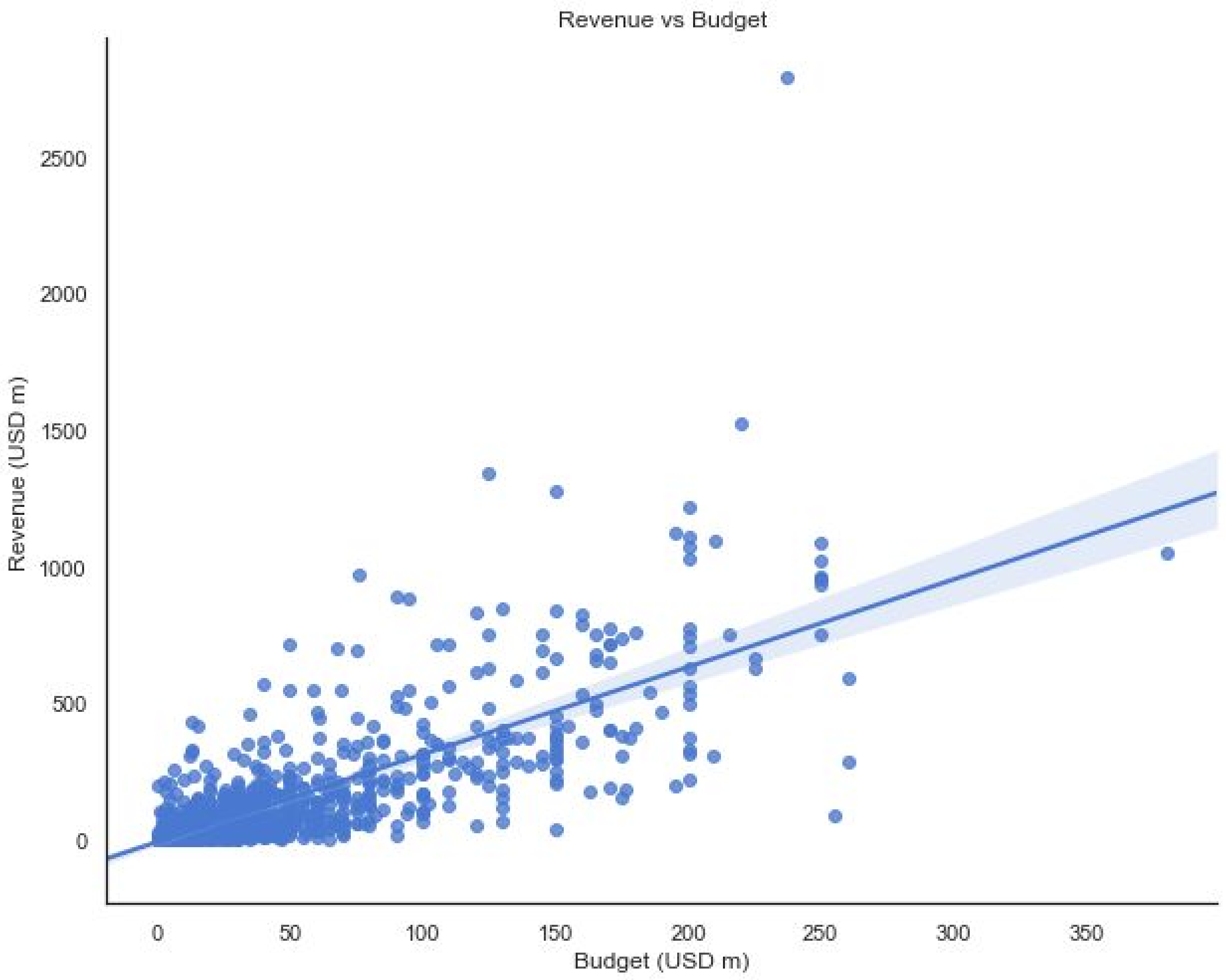
## Insights:

**Revenue and profit as proxies of popularity**

Profit appears to be very closely related to revenue. On further examination, we found that their correlation is 0.98. Given this close relationship, we will concentrate on revenue in later bivariate analysis, as profit would essentially have the same relationships.



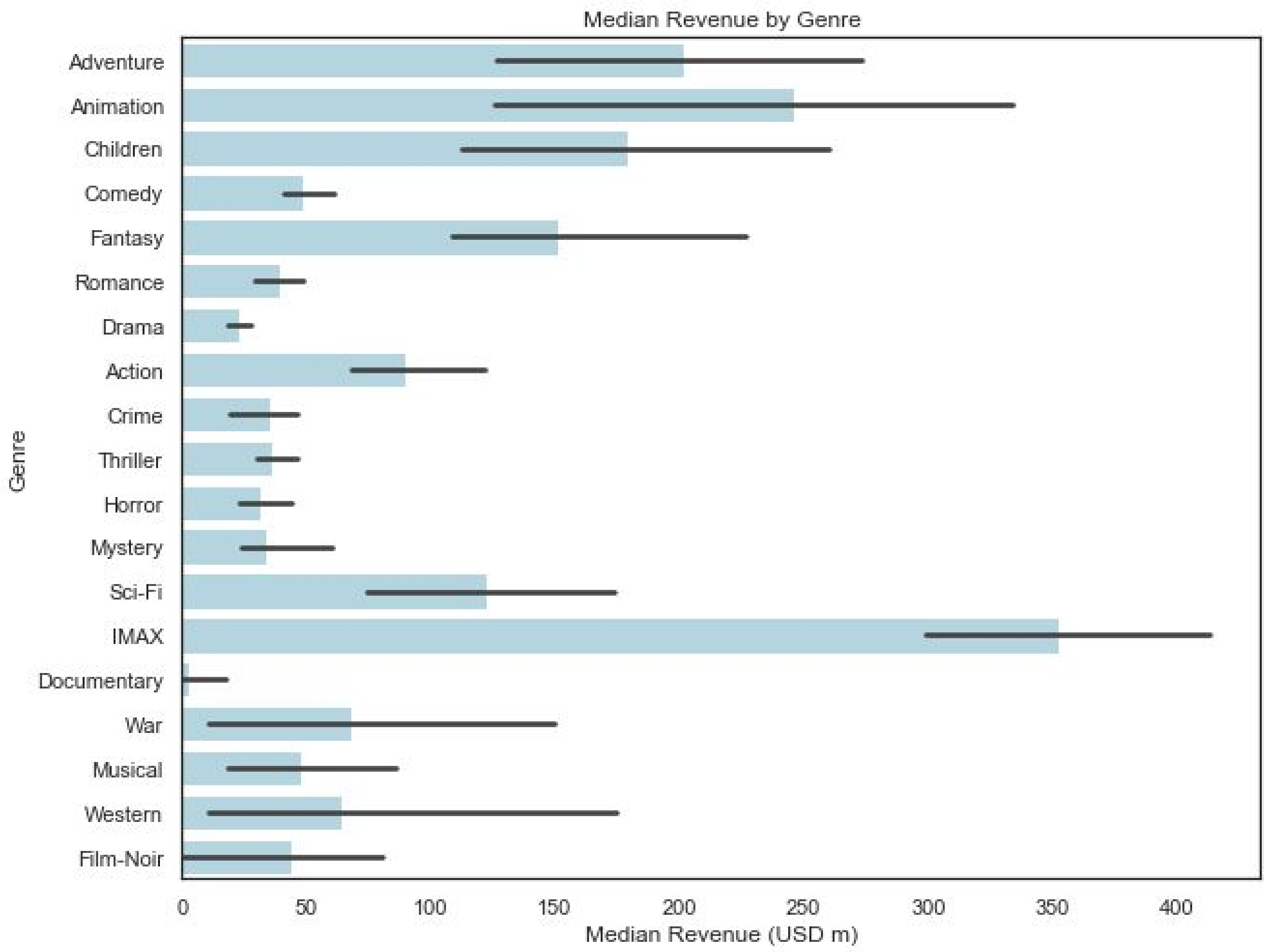
**Budget as a predictor of popularity**
Using revenue as a proxy for the popularity of a movie, we see a strong positive relationship between budget and revenue. Upon further analysis, we find a correlation of 0.78, a strong positive relationship, between the 2 features. We can conclude that budget should be a strong predictor of the popularity of a movie.



**Genre as a predictor of popularity**
Again we used revenue as a proxy for the popularity of a movie, using median revenue to avoid the positive skew that would be encountered with the mean. We find the rather strange result that IMAX movies have the highest median revenue, followed by animation, adventure, children, fantasy, sci-fi, and action.

The IMAX result may possibly be because only films that are confident of success would release in IMAX theatres. The other genres associated with high median revenues would be a good bets for recommendation.

However, we would need to know additional information about the user before recommending the children or animation genres.

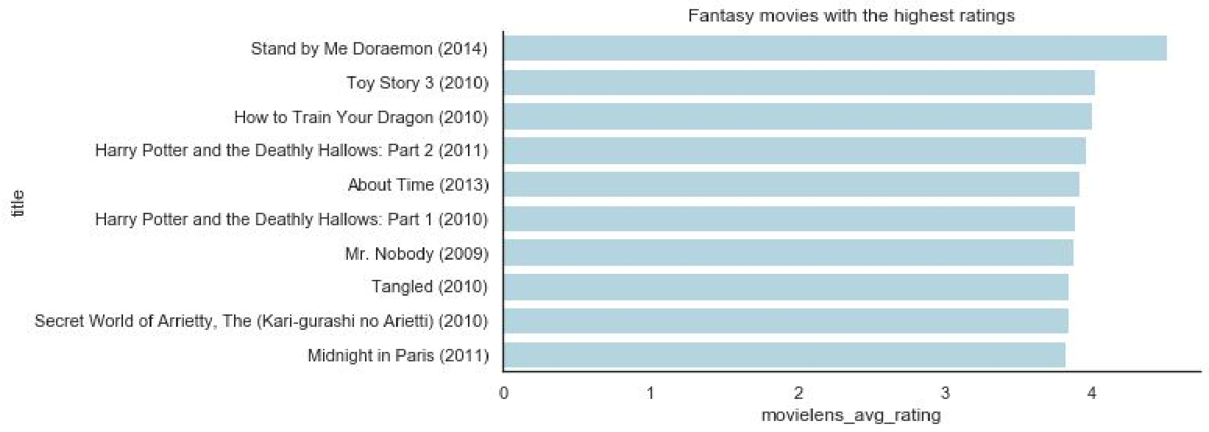


## Results:

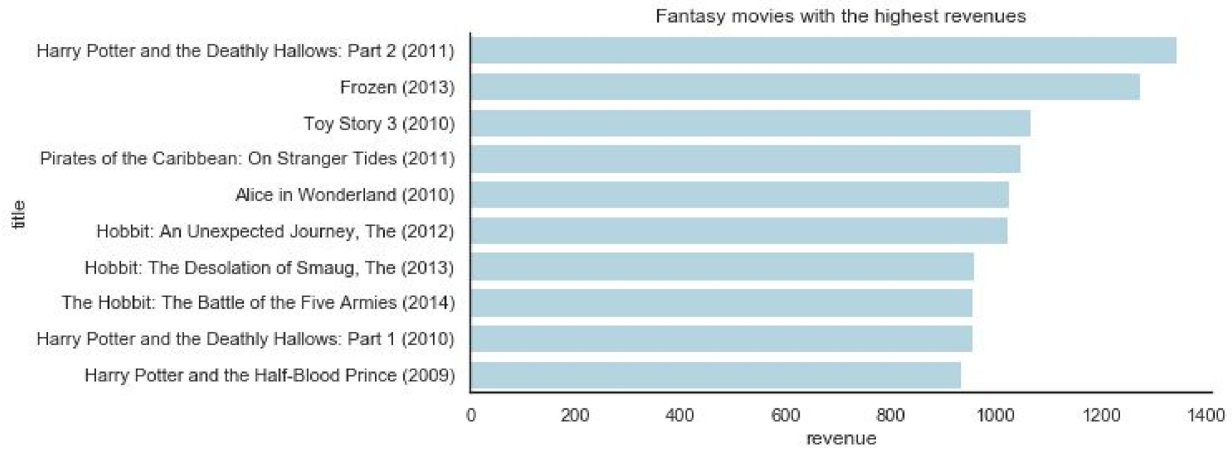
Given the information above, we can proceed to build a simple recommendation system.
The user would choose from a genre; then, for movies from the past, our program would simply provide the movies from that genre with a highest revenues or ratings.
For movies that are in cinemas, our program would provide recommendations based on the highest budgets, or by actors and directors associated with the highest ratings.

For example, if the user likes fantasy movies, the movies with the highest ratings of that type would be the following:

We note some obscure movies such as Doraemon that might not be a good recommendation for most users. This ranking is possibly due to a small number of users ranking this movie very highly.



However, by combining this with the following set of Fantasy movies with the highest revenues and finding the intersection



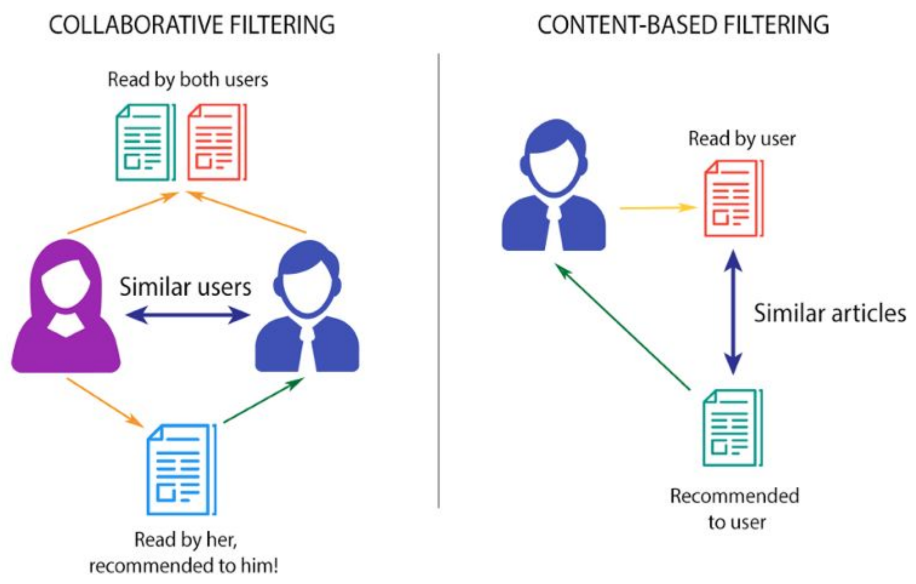
We would recommend the following movies for a fantasy lover:

Intersection of Fantasy movies with the highest revenues and ratings

| Title | Revenue (USD m) | Rating |
|---|---|---|
| Toy Story 3 (2010) | 1,066.96 | 4.012974 |
| Harry Potter and the Deathly Hallows: Part 2 (... | 1,342.00 | 3.950665 |
| Harry Potter and the Deathly Hallows: Part 1 (... | 954.30 | 3.879119 |
| Tangled (2010) | 591.79 | 3.841263 |
| Harry Potter and the Half-Blood Prince (2009) | 933.95 | 3.815312 |
| Hobbit: An Unexpected Journey, The (2012) | 1021.10 | 3.793884 |
| Hobbit: The Desolation of Smaug, The (2013) | 958.40 | 3.731915 |
| Frozen (2013) | 1274.21 | 3.706186 |

## Future analysis:

In order to build a more sophisticated recommendation engine, we would require more advanced statistical analysis as well as additional data.



Recommendation engine types

Content-based:
Using data directly provided by the user, the recommendation engine would calculate how similar items in its library are to the the items liked by the user, and recommend the most similar, improving with additional inputs from the user.

However, this method does not recommend items outside of what a user has tried, and does not leverage data from other users.

Collaborative-filtering:
On the other hand, we have access to granular ratings data from movielens, tagged to user ids. Using a collaborative filtering method, we would be able to calculate the similarity of a user to the user ids in our dataset, and recommend movies that similar user ids have enjoyed.