Analyzing the online popularity of movies

October 25, 2017

1 Project: Analyzing the online popularity of movies

1.1 Table of Contents

Introduction
The dataset in which this project is based on contains information about 10,000 movies collected from The Movie Database (TMDb). Among others, it includes data about the movies' popularity, budget, revenue, title, cast, runtime, genre, production, release date, online vote count and average.

The central questions we want to answer in this project are: - Is there a correlation between the offline popularity of a movie and its popularity in TMDb? - Which are the properties of the most high-rated movies in TMDb?
Data Wrangling

1.1.1 General Properties

```
In [57]: %matplotlib inline

         # Importing libraries
         import pandas as pd
         import numpy as np

In [17]: # Load the data
         movies = pd.read_csv('tmdb-movies.csv')

         # Print out the name of the columns
         print('Columns of the dataset')
         print()
         print(movies.columns)
         print()
         # Print out a few lines.
         movies.head(3)
```

```
Columns of the dataset

Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

Out[17]:

```
       id    imdb_id  popularity     budget     revenue      original_titl
0  135397  tt0369610   32.985763  150000000  1513528810      Jurassic Worl
1   76341  tt1392190   28.419936  150000000   378436354  Mad Max: Fury Roa
2  262500  tt2908446   13.112507  110000000   295238201           Insurger

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...

                                        homepage          director  \
0                  http://www.jurassicworld.com/   Colin Trevorrow
1                    http://www.madmaxmovie.com/     George Miller
2  http://www.thedivergentseries.movie/#insurgent  Robert Schwentke

                        tagline      ...       \
0            The park is open.      ...
1           What a Lovely Day.      ...
2  One Choice Can Destroy You      ...

                                            overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119

                                      genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller

                                production_companies release_date vote_cou
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       55
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       61
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       24

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
```

```
1          7.1          2015  1.379999e+08  3.481613e+08
2          6.3          2015  1.012000e+08  2.716190e+08

[3 rows x 21 columns]
```

In [18]:
```python
# Print out the number of rows and columns of the original dataset
print()
print("Properties of the original data frame. Rows: {0}, Columns: {1}".for
```

```
Properties of the original data frame. Rows: 10866, Columns: 21
```

1.1.2 Data Cleaning

In [87]:
```python
# Drop NA rows
movies = movies.dropna()

# Print out the number of rows after NA records were deleted
print()
print("Properties of the data frame after getting rid off NA. Rows: {0}, C
print()
# Select interested columns
movies_df = movies[['id', 'popularity', 'budget', 'revenue' , 'original_ti

# Print out the caracteristics of the dataframe after selecting the intere
print("Properties of the data frame after selecting the interested columns
```

```
Properties of the data frame after getting rid off NA. Rows: 1992, Columns: 21

Properties of the data frame after selecting the interested columns. Rows: 1992, Co
```

1.1.3 Data Transformation

In this section, I create extra variables to hold additional information about the movies.

Movies main genre

In [92]:
```python
# Create a column to hold the movies' main genre

def main_genre(movie):
    return movie.genres.split('|')[0]

movies_df = movies_df.assign(main_genre=movies_df.apply(main_genre, axis=1
movies_df.head(5)[['genres','main_genre']]
```

Out[92]:
```
                                          genres main_genre
0  Action|Adventure|Science Fiction|Thriller     Action
1  Action|Adventure|Science Fiction|Thriller     Action
2         Adventure|Science Fiction|Thriller  Adventure
3   Action|Adventure|Science Fiction|Fantasy     Action
4                      Action|Crime|Thriller     Action
```

Movies season release

In [93]:
```
# Create a column to hold the season (winter, summer, fall, spring) in whi
from datetime import datetime

def season_movie(movie):
    movie_rd = movie['release_date']
    rd = datetime.strptime(movie_rd, "%m/%d/%y")
    if rd.month in (12, 1, 2):
        return 'winter'
    elif rd.month in (3, 4, 5):
        return 'spring'
    elif rd.month in (6, 7, 8):
        return 'summer'
    elif rd.month in (9, 10, 11):
        return 'fall'
    else:
        return 'unknown'

movies_df = movies_df.assign(season=movies_df.apply(season_movie, axis=1))
movies_df.head(5)[['release_date','season']]
```

Out[93]:
```
  release_date  season
0       6/9/15  summer
1      5/13/15  spring
2      3/18/15  spring
3     12/15/15  winter
4       4/1/15  spring
```

Movies title length

In [94]:
```
# Create a column to hold the length of the movies' title

def title_length(movie):
    title = movie['original_title']
    return len(title)

movies_df = movies_df.assign(title_length=movies_df.apply(title_length, ax
movies_df.head(5)[['original_title','title_length']]
```

Out[94]:
```
                 original_title  title_length
0                Jurassic World            14
```

```
1            Mad Max: Fury Road          18
2                     Insurgent           9
3  Star Wars: The Force Awakens          28
4                     Furious 7           9
```

Exploratory Data Analysis

1.1.4 RQ1: Is there a correlation between the offline popularity of a movie and its popularity in TMDb?

To measure offline popularity, we will use the **revenue** of the movies as the proxy while the metric **popularity** measures the success of the movies in TMDb. Please refer here for more information about how TMDb builds the metric popularity.

```
In [79]: x = np.array(movies_df['revenue'])
         y = np.array(movies_df['popularity'])
         cor_x_y = np.corrcoef(x,y)[0][1]
         print('The correlation between popularity and revenue is: {0}'.format(cor_
```

```
The correlation between popularity and revenue is: 0.6413460877299713
```

Interestingly, the movies' popularity in TMDb show be highly and positively correlated with incomes of the movies (cor=0.64). Motivated by the previous result, next we are interested in understanding whether the movies' revenue is also correlated with the number of votes cast by the movie in TMDb. To answer the question, we check the correlation between **revenue** and **vote_count**.

```
In [80]: x = np.array(movies_df['revenue'])
         y = np.array(movies_df['vote_count'])
         cor_x_y = np.corrcoef(x,y)[0][1]
         print('The correlation between number of votes of the movies and their rev
```

```
The correlation between number of votes of the movies and their revenue is: 0.8047
```

1.1.5 RQ2: Which are the properties of the most high-rated movies in TMDb?

The rating of the movies will be measured through the **popularity** variable. For the properties of the movies, we will consider the following variables available in the dataset: **budget** and **runtime**. Also, we will use for this analysis the variables we generated before to the movies' **main_genre**, the **length** of their original title, and their **season** of release.

```
In [97]: # Selected the interested variables
         i_movies_df = movies_df[['popularity', 'budget', 'runtime', 'main_genre',
```

Present describe statistics for the entire dataset

```
In [98]: i_movies_df.describe()
```

```
Out[98]:        popularity        budget      runtime  title_length
         count  1992.000000  1.992000e+03  1992.000000   1992.000000
         mean      1.316763  3.454924e+07   106.040161     15.387550
         std       1.873563  5.061878e+07    29.234592      8.949628
         min       0.000620  0.000000e+00     0.000000      1.000000
         25%       0.384079  0.000000e+00    92.000000      9.000000
         50%       0.774223  1.500000e+07   102.000000     13.000000
         75%       1.538639  4.800000e+07   116.000000     19.000000
         max      32.985763  4.250000e+08   705.000000     83.000000
```

Show the top-5 main genres

```
In [107]: pd.crosstab(index=i_movies_df["main_genre"], columns="count").sort_values
```

```
Out[107]: col_0       count
          main_genre
          Drama         435
          Comedy        365
          Action        304
          Adventure     155
          Horror        150
```

Show the number of movies released per season

```
In [110]: pd.crosstab(index=i_movies_df["season"], columns="count").sort_values('co
```

```
Out[110]: col_0   count
          season
          fall      637
          summer    463
          spring    446
          winter    446
```

Split the dataset in to two groups: the top-50 high-rated movies and the rest

```
In [117]: # Sort the data from high to low popularity
          i_movies_df = i_movies_df.sort_values('popularity', ascending=False)
          i_50_movies_df = i_movies_df[:50]
          i_51_movies_df = i_movies_df[51:]
```

Show the average properties of both groups, the 50 most high-rated movies and the rest

```
In [179]: mean_50 = i_50_movies_df.describe().iloc[1].tolist()
          mean_51 = i_51_movies_df.describe().iloc[1].tolist()
          comparison = pd.DataFrame(data=[mean_50, mean_51],
```

```
                                  columns=['popularity', 'budget', 'runtime', 'ti
                                  index=['top-50', 'rest'])
         print(comparison)
```

```
       popularity        budget     runtime  title_length (num_char)
top-50   9.297445  1.254200e+08  133.020000                  18.8200
rest     1.108821  3.220354e+07  105.313756                  15.2983
```

Show the list of main genres in both groups

```
In [175]: main_genres_i50 = pd.crosstab(index=i_50_movies_df['main_genre'], columns
                            sort_values(by='count', ascending=False).head(5).to_dic
          main_genres_i51 = pd.crosstab(index=i_51_movies_df['main_genre'], columns
                            sort_values(by='count', ascending=False).head(5).to_dic
          comparison = pd.DataFrame(data=[main_genres_i50.keys(), main_genres_i51.k
                                  columns=['Rank1', 'Rank2', 'Rank3', 'Rank4', 'R
                                  index=['top-50', 'rest'])
          print(comparison)
```

```
         Rank1      Rank2   Rank3            Rank4      Rank5
top-50  Action  Adventure   Drama  Science Fiction    Fantasy
rest     Drama     Comedy  Action           Horror  Adventure
```

Show the number of movies per season in both groups

```
In [178]: seasons_i50 = pd.crosstab(index=i_50_movies_df['season'], columns='count'
                        sort_values(by='count', ascending=False).to_dict()['count']
          seasons_i51 = pd.crosstab(index=i_51_movies_df['season'], columns='count'
                        sort_values(by='count', ascending=False).to_dict()['count']
          comparison = pd.DataFrame(data=[seasons_i50.keys(), seasons_i51.keys()],
                                  columns=['Rank1', 'Rank2', 'Rank3', 'Rank4'],
                                  index=['top-50', 'rest'])
          print(comparison)
```

```
       Rank1   Rank2   Rank3   Rank4
top-50  fall  summer  winter  spring
rest    fall  summer  spring  winter
```

Conclusions

1.1.6 RQ1: Is there a correlation between the offline popularity of a movie and its popularity in TMDb?

We found that the movies' popularity in TMDb shows to be highly and positively correlated with incomes of the movies (cor=0.64). Moreover, we saw that the revenue of a movie is associated (cor=0.80) with the number of votes casted by the movie in TMDb. So, it seems that there is some sort of association between the success of movies in the box offices and their online valuation.

1.1.7 RQ2: Which are the properties of the most high-rated movies in TMDb?

Through the last analyses, we found some interesting properties that distinguished the top-50 high-rated movies from the rest. Fist, we saw that the top-50 high-rated movies invest more economic resources in their production than the rest. Also, we discovered that they are longer and named with longer titles. Besides, we found that their primary genres are, in general, different from the rest. The top-50 high-rated movies are mainly action and adventure films while the rest are more drama and comedy movies. However, we didn't find differences concerning the season release of the movies in both groups. In both cases, the periods of most premiers are fall and summer.

We haven't measure whether the discovered differences are significant, but from the results above there are some final comments we want to make. While it might be evident that expensive movies have higher chances of being successful, it is important to highlight the three aspects that characterize popular movies, which should be taken into account by producers. First, high-rated movies are longer, which seems to indicate that probably people like movies that spend more time in developing the story. Second, movies that attract the attention of the Internet users are usually those with high energy and exciting stories.