Investigate_a_Dataset

November 20, 2022

0.1 TMDb movie data

0.2 Table of Contents

1

1.1 Introduction

1.1.1 Dataset Description

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue. Certain columns, like 'cast' and 'genres', contain multiple values separated by pipe (|) characters. There are some odd characters in the 'cast' column. Don't worry about cleaning them. You can leave them as is. The final two columns ending with "_adj" show the budget and revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

1.1.2 Question(s) for Analysis

1- Which Genre Has The Highest Release Of Movies?
  2- Which year has the highest release of movies?
  3- Do most famous films have a long duration?

```
In [74]: # Use this cell to set up import statements for all of the packages that you
         #   plan to use.

         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         % matplotlib inline

         # Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

## Data Wrangling

1.1.3 General Properties

```
In [75]: # Load your data and print out a few lines. Perform operations to inspect data
         #   types and look for instances of missing or possibly errant data.
         df=pd.read_csv('tmdb-movies.csv')
         df.head()
```

```
Out[75]:        id    imdb_id  popularity     budget     revenue  \
         0  135397  tt0369610   32.985763  150000000  1513528810
         1   76341  tt1392190   28.419936  150000000   378436354
         2  262500  tt2908446   13.112507  110000000   295238201
         3  140607  tt2488496   11.173104  200000000  2068178225
         4  168259  tt2820852    9.335014  190000000  1506249360

                         original_title  \
         0                 Jurassic World
         1             Mad Max: Fury Road
         2                      Insurgent
         3  Star Wars: The Force Awakens
         4                      Furious 7

                                                         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...
         3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
         4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...

                                                     homepage          director  \
         0                      http://www.jurassicworld.com/   Colin Trevorrow
         1                        http://www.madmaxmovie.com/     George Miller
         2     http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
         3  http://www.starwars.com/films/star-wars-episod...       J.J. Abrams
         4                          http://www.furious7.com/         James Wan

                                tagline      ...       \
         0              The park is open.      ...
         1             What a Lovely Day.      ...
         2     One Choice Can Destroy You      ...
         3  Every generation has a story.      ...
         4            Vengeance Hits Home      ...

                                                  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
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137

                                      genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947

   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           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

```
In [76]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
```

```
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

```
In [77]: df.describe()
```

```
Out[77]:                   id    popularity        budget       revenue       runtime  \
         count   10866.000000  10866.000000  1.086600e+04  1.086600e+04  10866.000000
         mean    66064.177434      0.646441  1.462570e+07  3.982332e+07    102.070863
         std     92130.136561      1.000185  3.091321e+07  1.170035e+08     31.381405
         min         5.000000      0.000065  0.000000e+00  0.000000e+00      0.000000
         25%     10596.250000      0.207583  0.000000e+00  0.000000e+00     90.000000
         50%     20669.000000      0.383856  0.000000e+00  0.000000e+00     99.000000
         75%     75610.000000      0.713817  1.500000e+07  2.400000e+07    111.000000
         max    417859.000000     32.985763  4.250000e+08  2.781506e+09    900.000000

                  vote_count  vote_average  release_year    budget_adj   revenue_adj
         count  10866.000000  10866.000000  10866.000000  1.086600e+04  1.086600e+04
         mean     217.389748      5.974922   2001.322658  1.755104e+07  5.136436e+07
         std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
         min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
         25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
         50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
         75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
         max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

```
In [78]: #data has null values so we count total rows in each column which contain null values
         df.isnull().sum()
```

```
Out[78]: id                         0
         imdb_id                   10
         popularity                 0
         budget                     0
         revenue                    0
         original_title             0
         cast                      76
         homepage                7930
         director                  44
         tagline                 2824
         keywords                1493
         overview                   4
         runtime                    0
```

```
genres                      23
production_companies      1030
release_date                 0
vote_count                   0
vote_average                 0
release_year                 0
budget_adj                   0
revenue_adj                  0
dtype: int64
```

```
In [79]: #fill the null values with zero using 'fillna' function
         df1=df.fillna(0)
```

1.1.4 Data Cleaning

Removing Data (Duplicated and Unused information from the dataset)

2 1-remove duplicate rows from the dataset

```
In [12]:  #counting the duplicates

         sum(df1.duplicated())

Out[12]: 1
```

```
In [13]: #drop these duplicated rows
         df1.drop_duplicates(inplace=True)
         df1.shape

Out[13]: (10865, 21)
```

2.1 2- remove the unused colums.

```
In [16]: #The columns like imdb_id, homepage,tagline, overview, budget_adj and revenue_adj are n
         #I will drop these columns
         df1.drop(['imdb_id','homepage','tagline','overview','budget_adj','revenue_adj'],axis =1
```

```
In [17]: df1.shape

Out[17]: (10865, 15)
```

## Exploratory Data Analysis

2.1.1 Research Question 1 (Which Genre Has The Highest Release Of Movies?)

```
In [18]: #make a function will will split the string and return the count of each genre.
         def data(x):
             #concatenate all the rows of the genrs.
             data_plot = df[x].str.cat(sep = '|')
```

```
        data = pd.Series(data_plot.split('|'))
        #conts each of the genre and return.
        info = data.value_counts(ascending=False)
        return info
```

```
In [19]: total_genre_movies = data('genres')
         print(total_genre_movies)
```

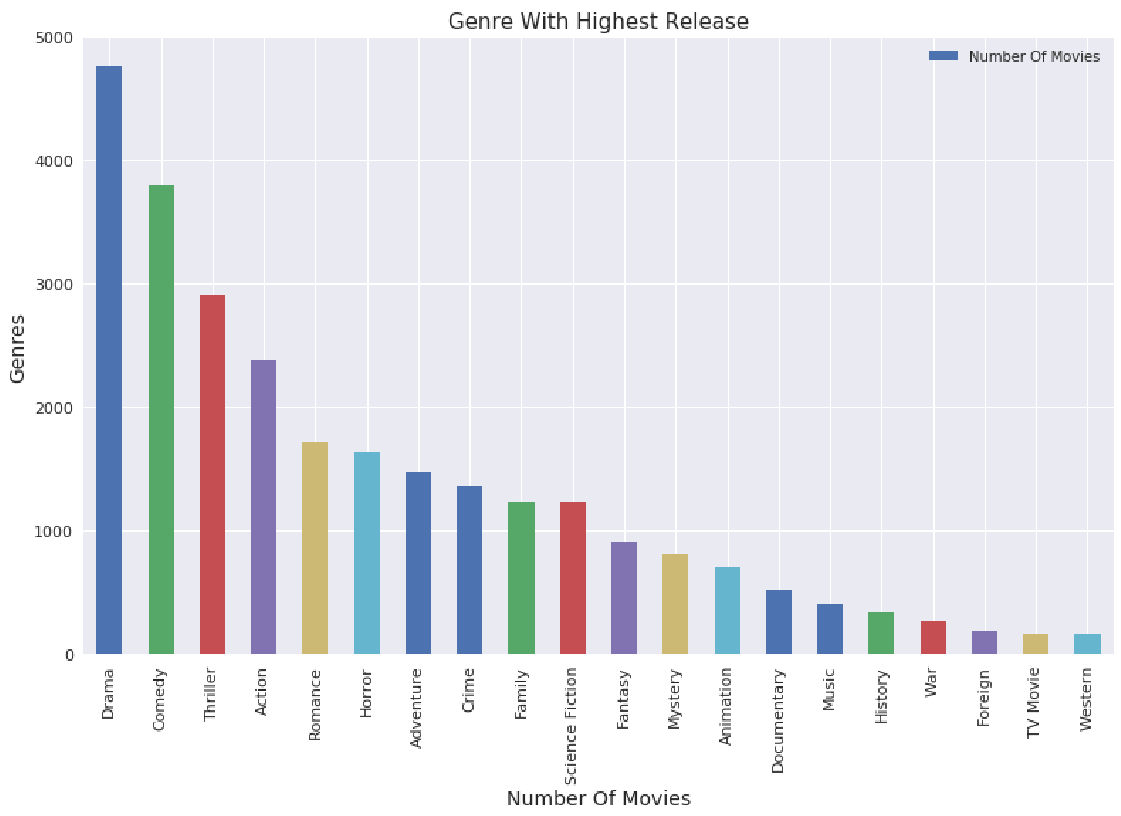
```
Drama              4761
Comedy             3793
Thriller           2908
Action             2385
Romance            1712
Horror             1637
Adventure          1471
Crime              1355
Family             1231
Science Fiction    1230
Fantasy             916
Mystery             810
Animation           699
Documentary         520
Music               408
History             334
War                 270
Foreign             188
TV Movie            167
Western             165
dtype: int64
```

```
In [73]: # plot a 'bar' plot using plot function for 'genre vs number of movies'.

         total_genre_movies.plot(kind= 'bar',figsize = (13,8),fontsize=11)

         #setup the title and the labels of the plot.

         plt.title("Genre With Highest Release",fontsize=15)
         plt.xlabel('Number Of Movies',fontsize=14)
         plt.ylabel('Genres',fontsize= 14)
         plt.legend(['Number Of Movies']);
```



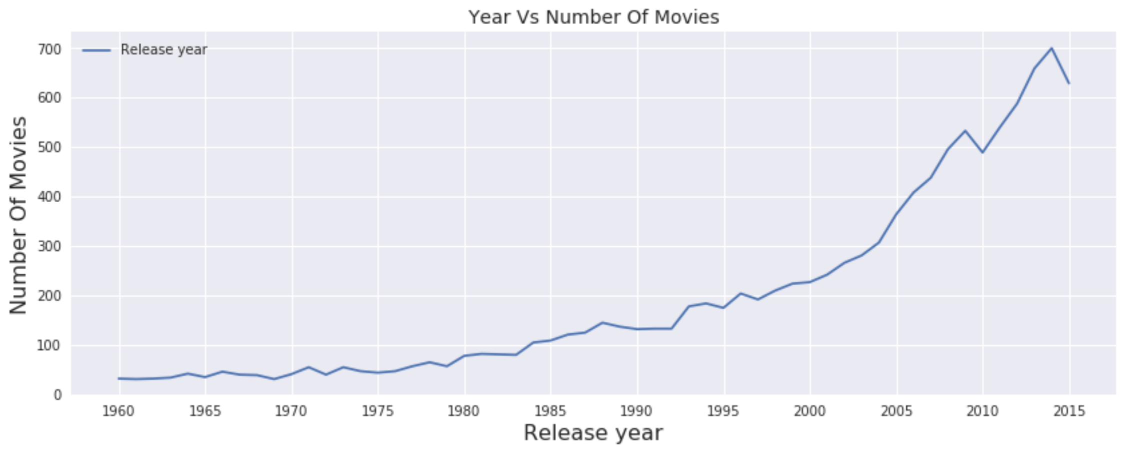
```
Here we can see the highest release of movies in all times chart show Drama movies are Most rel
```

2.2 Research Question 2 (Which year has the highest release of movies?)

```
In [21]: #count the number of movies in each year

         data=df1.groupby('release_year').count()['id']
```

```
In [63]: data.plot(xticks = np.arange(1960,2016,5))
         sns.set(rc={'figure.figsize':(14,5)})
         plt.title("Year Vs Number Of Movies",fontsize = 14)
         plt.xlabel('Release year',fontsize = 16)
         plt.ylabel('Number Of Movies',fontsize = 16)
         plt.legend(['Release year']);
```

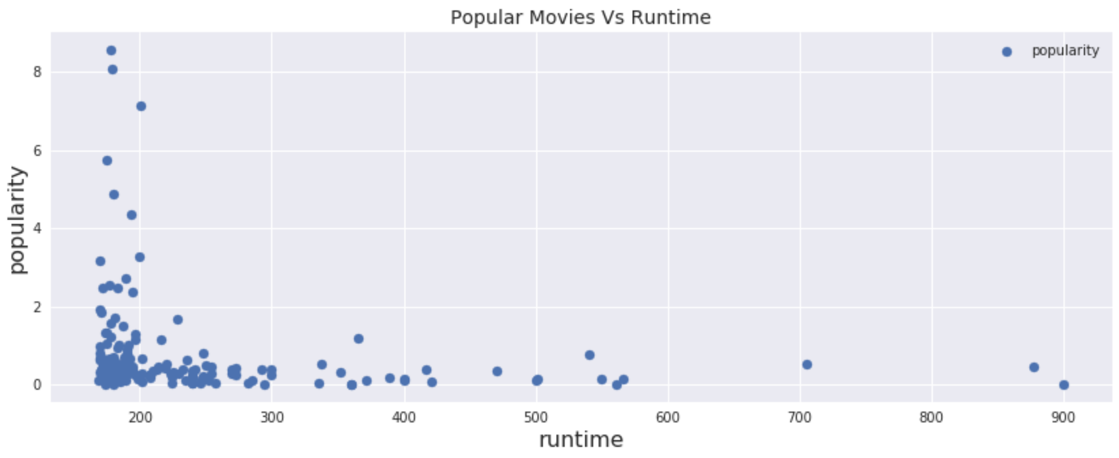


Here we can see The highest year in the release movies

3 Research Question 3 (Do most popular movies have a long duration? Let's find out)

```
In [68]: shorter_movies = df.sort_values(by=['runtime'], ascending = False).head(200)
         runtime = shorter_movies['runtime']
         popularity = shorter_movies['popularity']
```

```
In [71]: plt.scatter(runtime, popularity)
         plt.title("Popular Movies Vs Runtime",fontsize = 14)
         plt.xlabel('runtime',fontsize = 16)
         plt.ylabel('popularity',fontsize = 16)
         plt.legend(['popularity']);
```



```
we can see that the more popular movies is the shortest movies.
```

## Conclusions

In the first question, We will find through the analysis that the highest category in the issuance of films is the drama category, and then comes comedies, then horror films, and then comes the rest of the categories.

In second question,We find that the highest year in the release of films is the year 2015, and it comes in the second place with the highest release in 2010 and the third in 2005.

In third question, We find that the most popular films are in the short-term films compared to the long-term films

3.1 Limitations:

1- we are not sure if the data provided to us is completel corect and up-to-date.the budget and revenue column do not have currency unit 2- it might be possible different movies have budget in different currency according to the country they are produce in. So a disparity arises here which can state the complete analysis wrong. 3- i want to Drop the rows with missing values but it will affecte the overall analysis.During the data cleaning process

```
In [1]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])

Out[1]: 0
```