Investigate_a_Dataset

February 21, 2023

Project: Investigate a Dataset - [TMDb Movie data] Table of Contents Introduction Data Wrangling Exploratory Data Analysis Conclusions

Introduction

0.0.1 Dataset Description

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.This is a source to start asking in to those questions, with data on the plot, cast, crew, budget, and revenues of several thousand films Columns names:
id imdb_id popularity
budget
revenue original_title
cast
homepage
director
tagline overview
runtime genres
production_companies
release_date
vote_count
vote_average
release_year
budget_adj
revenue_adj

0.0.2 Question(s) for Analysis

what is the movie with highest revenue ? what is the movie with highest budget? what is the most popular genres? who is the actress with highest number of movies? is there relation between budget and revenue? is the budget spent on this industry differ from year to year? how is relation between revenue and popularity?

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        % matplotlib inline
```

```
In [3]: # Upgrade pandas to use dataframe.explode() function.
        !pip install --upgrade pandas==0.25.0
```

```
Requirement already up-to-date: pandas==0.25.0 in /opt/conda/lib/python3.6/site-packages (0.25.0
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in /opt/conda/lib/python3.6/site-
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in /opt/conda/lib/python
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /opt/conda/lib/python3.6/site-p
Requirement already satisfied, skipping upgrade: six>=1.5 in /opt/conda/lib/python3.6/site-packa
```

Data Wrangling

```
In [4]: df = pd.read_csv('tmdb-movies.csv')
        df.head()
```

Out[4]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams	Every generation has a story.	...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	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 [5]: df.shape
```

```
Out[5]: (10866, 21)
```

```
In [6]: df.describe()
```

```
Out[6]:                   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 [7]: 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 [8]: #check for null values
        df.isnull().sum()
```

```
Out[8]: 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 [9]:
```
#check for duplication
df.duplicated().sum()
```

Out[9]: 1

0.0.3 Data Cleaning

In [10]:
```
#Splitt genres  into separate columns
genres_df = df['genres'].str.split("|", expand=True)
genres_df.head(5)
```

Out[10]:

	0	1	2	3	4
0	Action	Adventure	Science Fiction	Thriller	None
1	Action	Adventure	Science Fiction	Thriller	None
2	Adventure	Science Fiction	Thriller	None	None
3	Action	Adventure	Science Fiction	Fantasy	None
4	Action	Crime	Thriller	None	None

In [11]:
```
# Creating a separate dataframe form unique genres records.
genres_df = genres_df.stack()
genres_df = pd.DataFrame(genres_df)
genres_df.head(5)
```

Out[11]:

		0
0	0	Action
	1	Adventure
	2	Science Fiction
	3	Thriller
1	0	Action

In [12]:
```
#Renaming the genres column and verifying the genres value count
genres_df.rename(columns={0:'genres_adj'}, inplace=True)
```

In [13]:
```
#Splitting the multiple cast entries into separate columns
cast_df = df['cast'].str.split("|", expand=True)
cast_df.head(5)
```

Out[13]:
```
                  0                    1                2  \
0       Chris Pratt  Bryce Dallas Howard       Irrfan Khan
1         Tom Hardy      Charlize Theron  Hugh Keays-Byrne
2  Shailene Woodley           Theo James      Kate Winslet
3     Harrison Ford          Mark Hamill     Carrie Fisher
4        Vin Diesel          Paul Walker     Jason Statham

                    3              4
0   Vincent D'Onofrio   Nick Robinson
1      Nicholas Hoult     Josh Helman
2        Ansel Elgort    Miles Teller
3         Adam Driver    Daisy Ridley
4  Michelle Rodriguez  Dwayne Johnson
```

In [14]:
```
# Creating a separate dataframe form unique cast records.
cast_df = cast_df.stack()
cast_df = pd.DataFrame(cast_df)
cast_df.head()
```

Out[14]:
```
                       0
0 0          Chris Pratt
  1  Bryce Dallas Howard
  2          Irrfan Khan
  3    Vincent D'Onofrio
  4        Nick Robinson
```

In [15]:
```
#Renaming the genres column and verifying the cast value count
cast_df.rename(columns={0:'cast_adj'}, inplace=True)
```

In [16]:
```
# drop duplicates in dataset
df.drop_duplicates(inplace=True)
```

In [17]:
```
#print number of duplicates again
df.duplicated().sum()
```

Out[17]:
```
0
```

In [18]:
```
# drop rows with any null values in both datasets
df.dropna(inplace=True)
```

In [19]:
```
#check again for null values
df.isnull().sum()
```

Out[19]:
```
id                      0
imdb_id                 0
popularity              0
budget                  0
revenue                 0
original_title          0
```

```
cast                    0
homepage                0
director                0
tagline                 0
keywords                0
overview                0
runtime                 0
genres                  0
production_companies    0
release_date            0
vote_count              0
vote_average            0
release_year            0
budget_adj              0
revenue_adj             0
dtype: int64
```

```
In [20]: #removing unnecessary coloumns
         df.drop(['id', 'imdb_id', 'cast', 'homepage', 'tagline', 'keywords', 'overview', 'runti
         df.head()
```

Out[20]:

	popularity	budget	revenue	original_title	director	release_year
0	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	2015
1	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	2015
2	13.112507	110000000	295238201	Insurgent	Robert Schwentke	2015
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	2015
4	9.335014	190000000	1506249360	Furious 7	James Wan	2015

Exploratory Data Analysis

what is the movie with highest revenue ?

what is the movie with highest budget?

```
In [21]: #column we should use it
         columns = ['budget','revenue']
```

```
In [22]: #function for our conclusions
         def highest_movie(columns):
             for column in columns:
                 max_fig = df[column].max()
                 max_indx = df[column].idxmax()
                 max_movie = df.loc[max_indx, 'original_title']
                 print("movie with highest {} is {} with {} by {}\n".format(column, max_movie, c
```

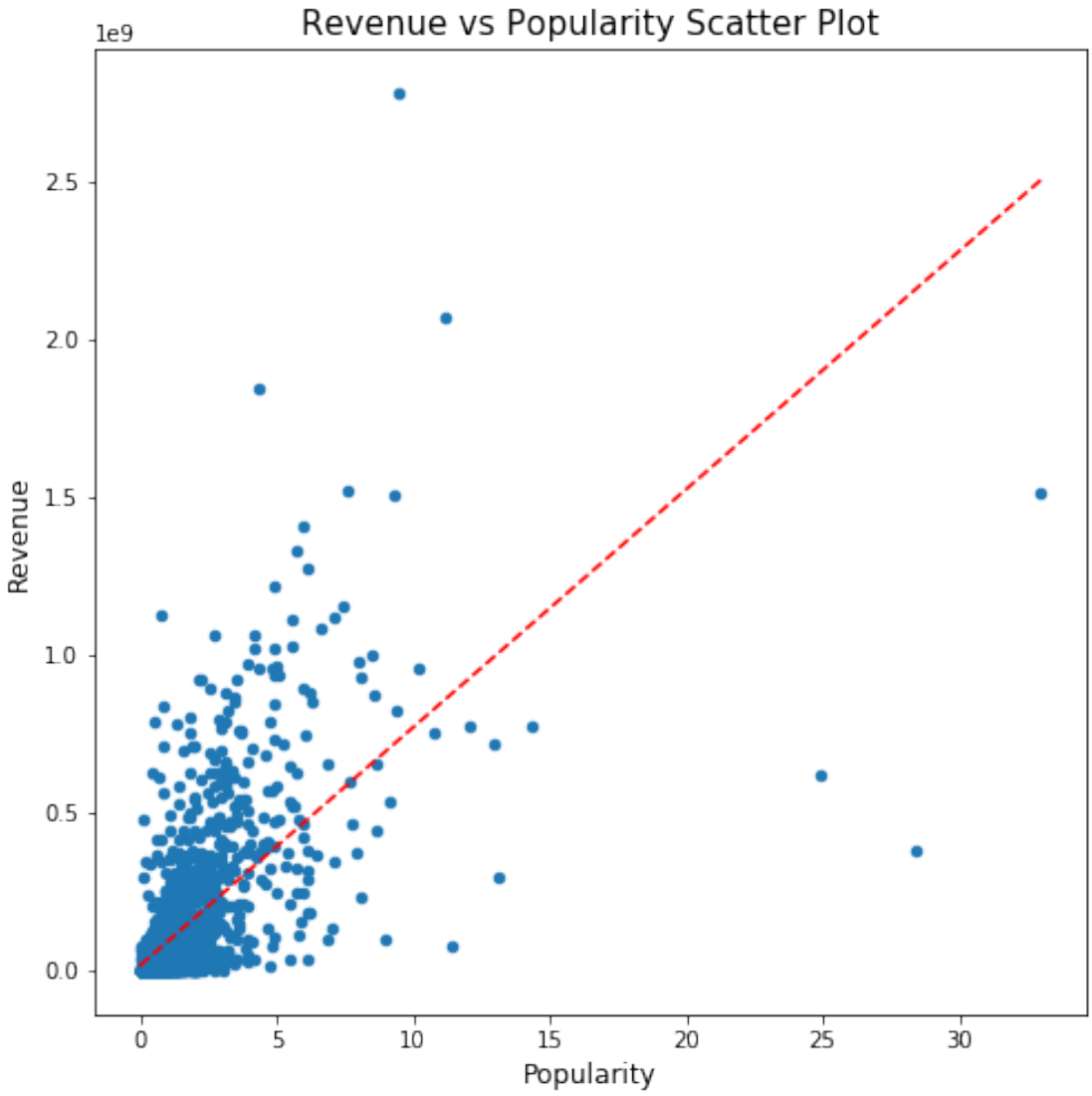
```
In [23]: highest_movie(columns)

movie with highest budget is The Warrior's Way with budget by 425000000

movie with highest revenue is Avatar with revenue by 2781505847
```

how is relation between revenue and popularity?

```
In [27]: #relation between revenue and popularity
         x='popularity'
         y='revenue'
         df.plot(x, y, kind='scatter', figsize=(8,8))
         plt.title('Revenue vs Popularity Scatter Plot', fontsize= 15)
         plt.xlabel('Popularity', fontsize= 12)
         plt.ylabel('Revenue', fontsize= 12)
         z = np.polyfit(df[x],df[y], 1)
         p = np.poly1d(z)
         plt.plot(df[x],p(df[x]),'r--');
```



there is a moderate relation between popularity and revenue
Relation between budget and revenue

```
In [ ]: df.plot(x='budget', y='revenue', kind='scatter')
```

is the budget spent on this industry differ from year to year?

```
In [ ]: df.plot(x='budget', y='release_year', kind='scatter')
```

what is the most popular genres?

```
In [ ]: genres_df.genres_adj.value_counts()
```

```
In [ ]: #visualisation for most popular genres
        genres_df.genres_adj.value_counts().plot(kind='pie', figsize= (8,8));
```

who is the actress with highest number of movies?

```
In [ ]: cast_df.cast_adj.value_counts()
```

Conclusions

-From this dataset we concluded that drama is the most genres followed by comedy then action. - Avatar is the movie with the highest revenue in this dataset. -The Warrior's Way is also the movie with the highest bunget in this dataset - we concluded also that Robert De Niro has highest number of movies followed by Samuel L. Jackson then Bruce Willis. -there is positive relation between budget and revenue. -this industry investment highly increased.

```
In [ ]: limitations:
            this dataset unfortunately has alot of missing data
            once we drop nan values we lose alot of data
            no currency specificatios
            dataset doesnt contain awards actors have won
```

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

```
In [ ]:
```