investigate-movie-dataset

October 19, 2018

1 Project: Investigate TMDb Movie Data

1.1 Table of Contents

This data set contains information about 10,000 movies released between 1960 and 2015. Collected from The Movie Database (TMDb), it also includes user ratings and popularities. This report aims at conducting exploratory data analysis to answer the questions below:

1. What are the average popularities of movies according to budget levels?
2. What are the profit trends of movies from year to year?
3. What are the average runtimes of movies over the years?
4. Which are the 5 cheapest and most expensive profitable movies of all time?

```
In [1]: # Use this cell to set up import statements for all of the packages that you
        #  plan to use.

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

        df = pd.read_csv('tmdb-movies.csv')
```

Data Wrangling

Key notes:: In this section of the report, the following work will be done: load the data; check for cleanliness; trim and clean dataset for analysis.

1.1.1 General Properties

In [2]: # Load your data and print out a few lines. Perform operations to inspect data

```
df.head()
```

Out[2]:
```
         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 [3]: # return a tuple of the dimensions of the dataframe

```
df.shape
```

Out[3]: (10866, 21)

In [4]: # print the column labels in the dataframe

```
for i, v in enumerate(df.columns):
    print(i, v)
```

```
0 id
1 imdb_id
2 popularity
3 budget
4 revenue
5 original_title
6 cast
7 homepage
8 director
9 tagline
10 keywords
11 overview
12 runtime
13 genres
14 production_companies
15 release_date
```

```
16 vote_count
17 vote_average
18 release_year
19 budget_adj
20 revenue_adj
```

In [5]: # return the datatypes of the columns

        df.dtypes

Out[5]: id   int64
        imdb_id   object
        popularity   float64
        budget   int64
        revenue   int64
        original_title   object
        cast   object
        homepage   object
        director   object
        tagline   object
        keywords   object
        overview   object
        runtime   int64
        genres   object
        production_companies   object
        release_date   object
        vote_count   int64
        vote_average   float64
        release_year   int64
        budget_adj   float64
        revenue_adj   float64
        dtype: object

In [6]: # check for duplicates in the data

        sum(df.duplicated())

Out[6]: 1

In [7]: # check if any value is NaN in DataFrame and in how many columns

        df.isnull().any().any(), sum(df.isnull().any())

Out[7]: (True, 9)

In [8]: # displays a concise summary of the dataframe
        # including the number of non-null values in each column

        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 [9]: *# Generates descriptive statistics, excluding NaN values*

      df.describe()

Out[9]:

	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
```

1.1.2 Data Cleaning

```
In [10]: # drop duplicates
         # confirm correction

         df.drop_duplicates(inplace=True)
         sum(df.duplicated())
```

```
Out[10]: 0
```

Firstly, we will get rid of the columns that are not needed for our analysis. They are id, imdb_id, budget_adj, revenue_adj, homepage, tagline, keywords and overview.

```
In [11]: # list of columns that are to be deleted/dropped
         col = ['id', 'imdb_id', 'budget_adj', 'revenue_adj', 'homepage', 'tagline', 'keyword

         # deleting the columns
         df.drop(col, axis = 1, inplace = True)

         #checking to see if the columns have been deleted
         df.head()
```

```
Out[11]:    popularity      budget      revenue              original_title  \
         0   32.985763  150000000  1513528810                 Jurassic World
         1   28.419936  150000000   378436354             Mad Max: Fury Road
         2   13.112507  110000000   295238201                      Insurgent
         3   11.173104  200000000  2068178225  Star Wars: The Force Awakens
         4    9.335014  190000000  1506249360                      Furious 7

                                                       cast         director  \
         0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...  Colin Trevorrow
         1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...     George Miller
         2  Shailene Woodley|Theo James|Kate Winslet|Ansel...  Robert Schwentke
         3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...       J.J. Abrams
         4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...          James Wan

            runtime                                   genres  \
         0      124  Action|Adventure|Science Fiction|Thriller
         1      120  Action|Adventure|Science Fiction|Thriller
         2      119          Adventure|Science Fiction|Thriller
         3      136    Action|Adventure|Science Fiction|Fantasy
         4      137                        Action|Crime|Thriller

                                production_companies release_date  vote_count  \
         0  Universal Studios|Amblin Entertainment|Legenda...      6/9/15        5562
```

6

```
    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
    0           6.5          2015
    1           7.1          2015
    2           6.3          2015
    3           7.5          2015
    4           7.3          2015
```

In [12]: # Changing datatype of `release_date` column
         df['release_date'] = pd.to_datetime(df['release_date'])

In [13]: # check if the change has taken place successfully
         df.dtypes

Out[13]: popularity   float64
         budget   int64
         revenue   int64
         original_title   object
         cast   object
         director   object
         runtime   int64
         genres   object
         production_companies   object
         release_date   datetime64[ns]
         vote_count   int64
         vote_average   float64
         release_year   int64
         dtype: object

In [14]: # Handling 0 values in `budget`, `revenue` and `runtime` columns
         # Making a list of the 3 columns
         temp_col = ['budget', 'revenue', 'runtime']

         # Replacing all the 0 values with NaN
         df[temp_col] = df[temp_col].replace(0, np.NAN)

In [15]: # Dropping/Deleting all the NaN values
         # Subset helps to define in which columns to look for missing values
         df.dropna(subset = temp_col, inplace = True)
         rows, col = df.shape

In [16]: #Changing format of budget and revenue column.

         change_type=['budget', 'revenue']
         #changing data type

```
df[change_type]=df[change_type].applymap(np.int64)
#printing the changed information
df.dtypes
```

Out[16]:
```
popularity                    float64
budget                          int64
revenue                         int64
original_title                 object
cast                           object
director                       object
runtime                       float64
genres                         object
production_companies           object
release_date            datetime64[ns]
vote_count                      int64
vote_average                  float64
release_year                    int64
dtype: object
```

1.2 Exploratory Data Analysis

Let's start with the exploration! :)

1.2.1   1. What are the average popularities of movies according to budget levels?

In [17]:
```python
# First we need to make columns for budget ranges
# We use the cut methods of the pandas library to do so
df['budget_ranges'] = pd.cut(df['budget'], df['budget'].describe()[3:8], labels = ['L
```

In [18]:
```python
# Now we find out the average popularities of each level
df.groupby('budget_ranges')['popularity'].mean()
```

Out[18]:
```
budget_ranges
Low                0.686413
Medium             0.951718
Moderately High    1.142414
High               2.080911
Name: popularity, dtype: float64
```

In [19]:
```python
# Plotting the above information in a bar graph
sns.set()
df.groupby('budget_ranges')['popularity'].mean().plot(kind = 'bar', figsize = (16, 8))

# Setting the title of the plot
plt.title('Average popularities of movies by budget levels', fontsize = 18)

# Setting the x and y axis labels
plt.xlabel('Budget Ranges', fontsize = 16)
plt.ylabel('Popularity', fontsize = 16);
```

Average popularities of movies by budget levels
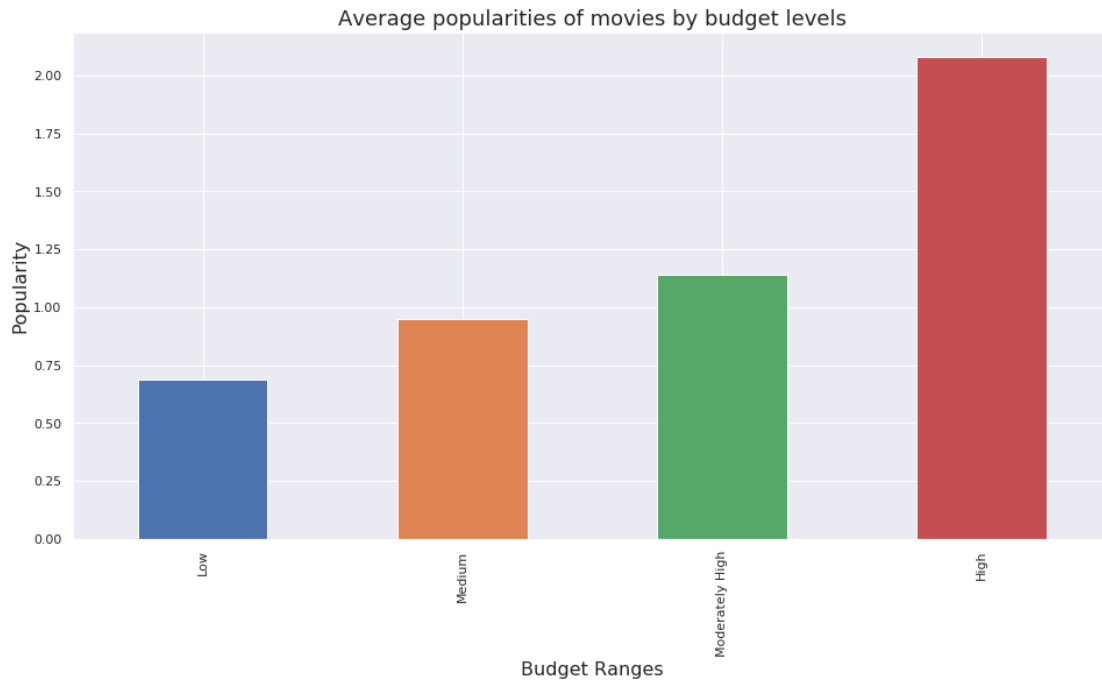
It appear that movies with a highter budget are more popular.

1.2.2 2. What are profit trends from year to year?

```
In [20]: # First we need to insert a column for the profit/loss value of each movie
         df.insert(3, 'profit_loss', df['revenue'] - df['budget'])

In [21]: df.groupby('release_year')['profit_loss'].sum().plot(kind = 'line', figsize = (16, 8)
         plt.title('Profit Trends from year to year', fontsize = 18)
         plt.xlabel('Year', fontsize = 16)
         plt.ylabel('Profit', fontsize = 16);
```
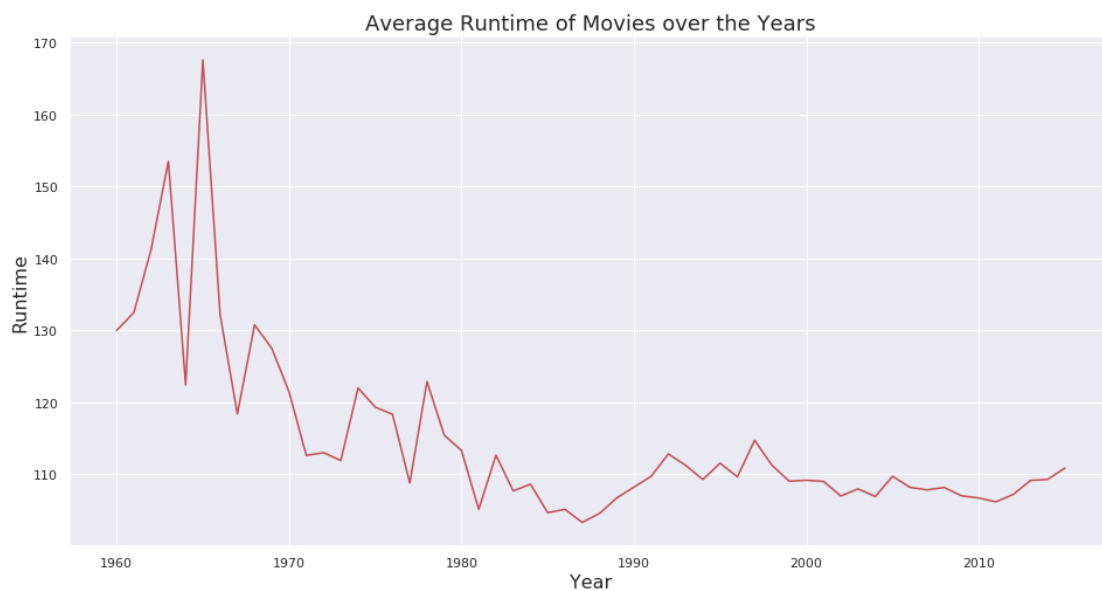
Profits increased over the years.

1.2.3 3. What are the average runtimes of movies over the years?

```
In [22]: df.groupby('release_year')['runtime'].mean().plot(kind='line', figsize = (16, 8), col
         plt.title('Average Runtime of Movies over the Years', fontsize = 18)
         plt.xlabel('Year', fontsize = 16)
         plt.ylabel('Runtime', fontsize = 16);
```

The runtime of movies has decreased over the years.

1.2.4 4. Which are the 5 cheapest and most expensive profitable movies of all time?

For this we shall set a standard value of profit which has to be met, that value will be $50,000,000

```
In [23]: # creating a list of columns that will be viewed
         col = ['original_title', 'cast', 'director', 'budget', 'revenue', 'profit_loss']

         # Using query function to show records of movies which have a profit of more than $50
         # Also using sort_values function to make sure it is sorted according to the budget c

         df.query('profit_loss>50000000')[col].sort_values('budget', ascending = False).head(5)
```

```
Out[23]:                                  original_title  \
         3375  Pirates of the Caribbean: On Stranger Tides
         7387      Pirates of the Caribbean: At World's End
         14                          Avengers: Age of Ultron
         6570                            Superman Returns
         1929                                     Tangled

                                                     cast  \
         3375  Johnny Depp|PenÃ¯lope Cruz|Geoffrey Rush|Ian M...
         7387  Johnny Depp|Orlando Bloom|Keira Knightley|Geof...
         14     Robert Downey Jr.|Chris Hemsworth|Mark Ruffalo...
         6570  Brandon Routh|Kevin Spacey|Kate Bosworth|James...
         1929  Zachary Levi|Mandy Moore|Donna Murphy|Ron Perl...

                              director      budget      revenue  profit_loss
         3375            Rob Marshall   380000000   1021683000    641683000
         7387           Gore Verbinski   300000000    961000000    661000000
         14                Joss Whedon   280000000   1405035767   1125035767
         6570             Bryan Singer   270000000    391081192    121081192
         1929  Nathan Greno|Byron Howard   260000000    591794936    331794936
```

```
In [24]: df.query('profit_loss>50000000')[col].sort_values('budget', ascending = True).head(5)
```

```
Out[24]:               original_title  \
         10495  The Karate Kid, Part II
         7447        Paranormal Activity
         2449    The Blair Witch Project
         7057                 Open Water
         10759                 Halloween

                                                     cast  \
         10495  Ralph Macchio|Pat Morita|Martin Kove|Charlie T...
         7447    Katie Featherston|Micah Sloat|Mark Fredrichs|A...
         2449    Heather Donahue|Michael C. Williams|Joshua Leo...
```

11

```
7057    Blanchard Ryan|Daniel Travis|Saul Stein|Michae...
10759   Donald Pleasence|Jamie Lee Curtis|P.J. Soles|N...


                             director  budget     revenue  profit_loss
10495              John G. Avildsen       113   115103979    115103866
7447                     Oren Peli      15000   193355800    193340800
2449   Daniel Myrick|Eduardo SÃą̊nchez  25000   248000000    247975000
7057                   Chris Kentis    130000    54667954     54537954
10759               John Carpenter    300000    70000000     69700000
```

Conclusions

Question 1: It can be observed that movies with a higher budget range tend to be more popular with the audience.

Question 2: Profits have increased exponentially with each passing year especially after the beginning of the 21st century.

Question 3: The runtime of movies has decreased with each passsing year. It experienced a hike during the 60s but has then steadily decreased over the years. The lowest was around 100-110 mintues. Presently, movies tend to last around the 110 minute mark.

Limitations: This analysis was done considering the movies which had a significant amount of profit of around 50 million dollar. This might not be completely error free but by following these suggestion one can increase the probability of a movie to become a hit. Moreover we are not sure if the data provided to us is completel corect and up-to-date. As mentioned before the budget and revenue column do not have currency unit, 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. Dropping the rows with missing values also affected the overall analysis.