report

September 29, 2018

1 Machine Learning Foundation Nanodegree

2 Project: Investigate a TMDb movie Database

##
Table
of
Contents

- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**

2.1 Introduction

In this project of my Data Analysis, I am investigating a TMDb movies database file which has collection of important detials of about 10k+ movies, including their details of budget, revenue, release dates, etc.

Let's take a glimpse at TMDb movie database csv file...

```
In [1]: import pandas as pd

        #reading tmdb csv file and storing that to a variable
        glimpse_tmdb = pd.read_csv('data.csv')

        #calling out first 5 rows (excluding headers) of tmdb database
        glimpse_tmdb.head()
```

Out[1]:

	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		...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan		...

```
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...     6/9/2015       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...    5/13/2015       6185
2  Summit Entertainment|Mandeville Films|Red Wago...    3/18/2015       2480
3          Lucasfilm|Truenorth Productions|Bad Robot   12/15/2015       5292
4  Universal Pictures|Original Film|Media Rights ...     4/1/2015       2947

   vote_average  release_year   budget_adj   revenue_adj
0           6.5          2015  137999939.3  1.392446e+09
1           7.1          2015  137999939.3  3.481613e+08
2           6.3          2015  101199955.5  2.716190e+08
3           7.5          2015  183999919.0  1.902723e+09
4           7.3          2015  174799923.1  1.385749e+09

[5 rows x 21 columns]
```

2.1.1 What can we say about the dataset provided?

The columns *'budget', 'revenue', 'budget_adj', 'revenue_adj'* has not given us the currency but for this dataset we will assume that it is in dollars.

The vote count for each movie is not similar, for example, the movie *'Mad Max : Fury Road'* has *6k+* votes while *Sinister 2* has only *331 votes* (as seen above). Since the votes of the movies vary so much the *vote_average* column also is effected by it. So we cannot calculate or assume that movie with highest votes or rating was more successful since the voters of each film vary.

2.1.2 What Questions can be brainstormed?

Looking at this database...

The first question comes in my mind is which movie gained the most profit or we can also kind of say that which movie has been the people's favourite?

Since this is just the glimpse of the database, the glimpse of the data just shows the movies in the year 2015, but there are also other movies released in different years so the Second question comes in my mind is in which year the movies made the most profit?

Finally my curious mind wanted to know what are the similar characteristics of movies which have gained highest profits?

2.1.3 What needs to be Wrangled and Cleaned

Based on the questions brainstormed above, we want to know do we have all the valid values of the variables that we want to calculate and how can this data be trimmed so we can only have the columns we need. This will also make our dataset clean and easy for us to calculate what we want.

As you can see in this database of movies there are lots of movies where the budget or revenue have a value of '0' which means that the values of those variables of those movies has not been recorded. Calculating the profits of these movies would lead to inappropriate results. So we need to delete these rows.

Also this dataset has some duplicate rows. We have to clean that too for appropriate results.

We will also calculate the average runtime of the movies so in case if we have a runtime of a movie '0' then we need to replace it with `NaN`.

The *'release_date'* column must be converted into date format.

Checking if all columns are in the desired data type, if not then we have to change it.

Mentioning the country currency in the desired columns.

Finally, we will also remove unnecessory columns such as *'id'*, *'imdb_id'*, *'popularity'*, *'budget_adj'*, *'revenue_adj'*, *'homepage'*, *'keywords'*, *'overview'*, *'production_companies'*, *'vote_count'* and *'vote_average'*.

2.1.4 Questions to be Answered

General questions about the dataset.

```
<ol type = 'a'>
        <li>Which movie earns the most and least profit?</li>
        <li>Which movie had the greatest and least runtime?</li>
        <li>Which movie had the greatest and least budget?</li>
        <li>Which movie had the greatest and least revenue?</li>
        <li>What is the average runtime of all movies?</li>
        <li>In which year we had the most movies making profits?</li>
    </ol>
<li>What are the similar characteristics does the most profitable movie have?</li>
    <ol type = 'a'>
        <li>Average duration of movies.</li>
        <li>Average Budget.</li>
        <li>Average revenue.</li>
        <li>Average profits.</li>
        <li>Which director directed most films?</li>
        <li>Whcih cast has appeared the most?</li>
        <li>Which genre were more successful?</li>
        <li>Which month released highest number of movies in all of the years? And which month
    </ol>
```

2.2 Data Cleaning

Before answering the above questions we need a clean dataset which has columns and rows we need for calculations.

First, lets clean up the columns. We will only keep the columns we need and remove the rest of them.

Columns to delete - `id`, `imdb_id`, `popularity`, `budget_adj`, `revenue_adj`, `homepage`, `keywords`, `overview`, `production_companies`, `vote_count` and `vote_average`.

```
In [2]: #importing all the nescessory libraries we need for our analysis
        import numpy as np
        from matplotlib import pyplot as plt
        import seaborn as sns

        #this variable will store the database of tmdb movies into a dataframe
        movie_data = pd.read_csv('data.csv')
```

Let's see how many entries we have of movies in this dataset and features.

```
In [3]: rows, col = movie_data.shape
        #since 'rows' includes count of a header, we need to remove its count.
        print('We have {} total entries of movies and {} columns/features of it.'.format(rows-
```

```
We have 10865 total entries of movies and 21 columns/features of it.
```

```
In [4]: #lets give a list of movies that needs to be deleted
        del_col = [ 'id', 'imdb_id', 'popularity', 'budget_adj', 'revenue_adj', 'homepage', 'k


        #deleting the columns from the database
        movie_data = movie_data.drop(del_col, 1)
        #now take a look at this new dataset
        movie_data.head(3)
```

Out[4]:

	budget	revenue	original_title	cast	director	tagline	runtime
0	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	124
1	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller		
2	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke		

```
1          What a Lovely Day.      120
2  One Choice Can Destroy You      119

                                        genres release_date  release_year
0  Action|Adventure|Science Fiction|Thriller     6/9/2015          2015
1  Action|Adventure|Science Fiction|Thriller    5/13/2015          2015
2         Adventure|Science Fiction|Thriller    3/18/2015          2015
```

The difference between the database before and now can be seen. This database is soothing to eyes as it only contains columns that are needed for analysis.

Now lets clean for any duplicate rows.

```
In [5]: #will drop duplicate rows but will keep the first one
        movie_data.drop_duplicates(keep = 'first', inplace = True)

        rows, col = movie_data.shape
        print('We now have {} total entries of movies and {} columns/features of it.'.format(r

We now have 10864 total entries of movies and 10 columns/features of it.
```

So we had one duplicate copy of a movie. Now we have 10864 movie entries.

Now, lets figure out which movies have a value of `'0'` in their `budget` or `revenue`, and then deleting those movies from database.

```
In [6]: #giving list of column names that needs to be checked
        check_row = ['budget', 'revenue']

        #this will replace the value of '0' to NaN of columns given in the list
        movie_data[check_row] = movie_data[check_row].replace(0, np.NaN)

        #now we will drop any row which has NaN values in any of the column of the list (check_

        movie_data.dropna(subset = check_row, inplace = True)

        rows, col = movie_data.shape
        print('After cleaning, we now have only {} entries of movies.'.format(rows-1))

After cleaning, we now have only 3853 entries of movies.
```

As you saw in the previous dataset from having 10k+ rows and 21 columns we have now come down to 3853 rows and 10 columns. These many columns are needed for analysis and we have all the rows that have valid values for our calculations.

Now as we are done with cleaning the dataset, let's move on to data wrangling phase.

2.3 Data Wrangling

Now first lets check if we have any movie with a runtime value of `0`. If we have any, we will replace with `NaN`.

```
In [7]: #replacing 0 with NaN of runtime column of the dataframe
        movie_data['runtime'] = movie_data['runtime'].replace(0, np.NaN)
```

Now we need to convert the `'release_date'` column to date format

```
In [8]: #calling the column which need to be formatted in datetime and storing those values in
        movie_data.release_date = pd.to_datetime(movie_data['release_date'])

        #showing the dataset
        movie_data.head(2)
```

```
Out[8]:         budget       revenue      original_title  \
        0  150000000.0  1.513529e+09      Jurassic World
        1  150000000.0  3.784364e+08  Mad Max: Fury Road

                                                  cast         director  \
        0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...  Colin Trevorrow
        1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...    George Miller

                    tagline  runtime                                     genres  \
        0   The park is open.      124  Action|Adventure|Science Fiction|Thriller
        1  What a Lovely Day.      120  Action|Adventure|Science Fiction|Thriller

          release_date  release_year
        0   2015-06-09          2015
        1   2015-05-13          2015
```

As you see, the `'release_date'` column has been changed to date format. (year-month-day)
Lets see if all the columns are in the format that we want for our calculations.

```
In [9]: #shwoing the datatypes of all the columns
        movie_data.dtypes
```

```
Out[9]: budget                   float64
        revenue                  float64
        original_title            object
        cast                      object
        director                  object
        tagline                   object
        runtime                    int64
        genres                    object
        release_date      datetime64[ns]
        release_year               int64
        dtype: object
```

As we can see we have float values for `'budget'` and `'revenue'` columns, since we dont need float but in int datatype, lets convert them.

```
In [10]: #applymap function changes the columns data type to the type 'argument' we pass
         change_coltype = ['budget', 'revenue']

         movie_data[change_coltype] = movie_data[change_coltype].applymap(np.int64)
         #shwoing the datatypes of all columns
         movie_data.dtypes
```

```
Out[10]: budget                    int64
         revenue                   int64
         original_title           object
         cast                     object
         director                 object
         tagline                  object
         runtime                   int64
         genres                   object
         release_date      datetime64[ns]
         release_year              int64
         dtype: object
```

Now all columns are in the desired format.

Since the values in the column `'budget'` and `'revenue'` shows us in Currency of US (as assumed earlier), lets change the name of these columns for convenience.

```
In [11]: #rename function renames the columns, the key as being the old name and its value new
         movie_data.rename(columns = {'budget' : 'budget_(in_US-Dollars)', 'revenue' : 'revenue
```

Since now we have the columns, rows and format of the dataset in right way, its time to investigate the data for the questions asked.

2.4 Exploratory Data Analysis

Before answering the questions, lets figure out the profits of each movie.

```
In [12]: #assigning a new column which will hold the profit values of each movie

         #the insert function's first argument is an index number given to locate the column,
         #...and last but not least it takes the calculation values to output for specific col

         #To calculate profit of each movie, we need to substract the budget from the revenue
         movie_data.insert(2, 'profit_(in_US_Dollars)', movie_data['revenue_(in_US-Dollars)'] -

         #for just in case situations or for convenience, we change the data type to int
         movie_data['profit_(in_US_Dollars)'] = movie_data['profit_(in_US_Dollars)'].apply(np.i

         #showing the dataset
         movie_data.head(2)
```