Investigate_a_Dataset

October 16, 2018

1 Project: Investigate a Movies Dataset

1.1 Table of Contents

Introduction
    Data Wrangling
    Exploratory Data Analysis
    Conclusions
    ## Introduction Investigate a movie dataset to find the common thing for a good movie as well as some of their charateristic to be able to make the most profit

```
In [136]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
          % matplotlib inline
```

    ## Data Wrangling

1.1.1 General Properties

```
In [137]: df = pd.read_csv('tmdb-movies.csv')
          df.head()
```

```
Out[137]:        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 [138]: df.shape
```

```
Out[138]: (10866, 21)
```

```
In [139]: df.describe()
```

```
Out[139]:                  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 [ ]:
```

```
In [140]: 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
```

1.1.2 Data Cleaning (Replace this with more specific notes!)

```
In [141]: df.drop(['id', 'imdb_id', 'original_title', 'cast', 'homepage', 'director', 'tagline',
```

```
In [142]: df.head()
```

Out[142]:

	popularity	budget	revenue	runtime	genres	vote_count	vote_average	release_year
0	32.985763	150000000	1513528810	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015
1	28.419936	150000000	378436354	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015
2	13.112507	110000000	295238201	119	Adventure|Science Fiction|Thriller	2480	6.3	2015
3	11.173104	200000000	2068178225	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015
4	9.335014	190000000	1506249360	137	Action|Crime|Thriller	2947	7.3	2015

```
In [143]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 8 columns):
popularity      10866 non-null float64
budget          10866 non-null int64
revenue         10866 non-null int64
runtime         10866 non-null int64
genres          10843 non-null object
```

```
vote_count      10866 non-null int64
vote_average    10866 non-null float64
release_year    10866 non-null int64
dtypes: float64(2), int64(5), object(1)
memory usage: 679.2+ KB
```

```
In [144]: df[df.genres.isnull()]
```

Out[144]:

	popularity	budget	revenue	runtime	genres	vote_count	vote_average	release_year
424	0.244648	0	0	100	NaN	21	6.1	2015
620	0.129696	0	0	90	NaN	13	5.0	2015
997	0.330431	0	0	44	NaN	13	6.8	2014
1712	0.302095	0	0	88	NaN	57	7.4	2009
1897	0.020701	0	0	76	NaN	11	7.0	2009
2370	0.081892	0	0	0	NaN	12	5.8	2010
2376	0.068411	0	0	62	NaN	11	7.7	2010
2853	0.130018	0	0	110	NaN	12	7.2	2001
3279	0.145331	0	0	96	NaN	11	6.1	2008
4547	0.520520	0	0	220	NaN	12	8.3	2012
4732	0.235911	0	0	100	NaN	12	6.2	2012
4797	0.167501	0	0	60	NaN	10	7.8	2012
4890	0.083202	0	0	2	NaN	14	7.0	2012
5830	0.248944	0	0	60	NaN	26	8.5	2015
5934	0.067433	0	0	3	NaN	27	6.9	2015
6043	0.039080	0	0	127	NaN	12	5.9	2015
6530	0.092724	0	0	6	NaN	24	5.9	2015
8234	0.028874	0	0	103	NaN	44	6.7	2015
8614	0.273934	0	0	12	NaN	14	6.7	2015
8878	0.038045	0	0	85	NaN	16	5.4	2015
9307	0.094652	0	0	105	NaN	10	5.3	2015
9799	0.175008	0	0	5	NaN	11	5.0	2015
10659	0.344172	5000	0	71	NaN	10	3.0	2015

```
5830          2013
5934          2013
6043          2013
6530          2005
8234          1995
8614          1996
8878          2000
9307          1989
9799          1974
10659         1970
```

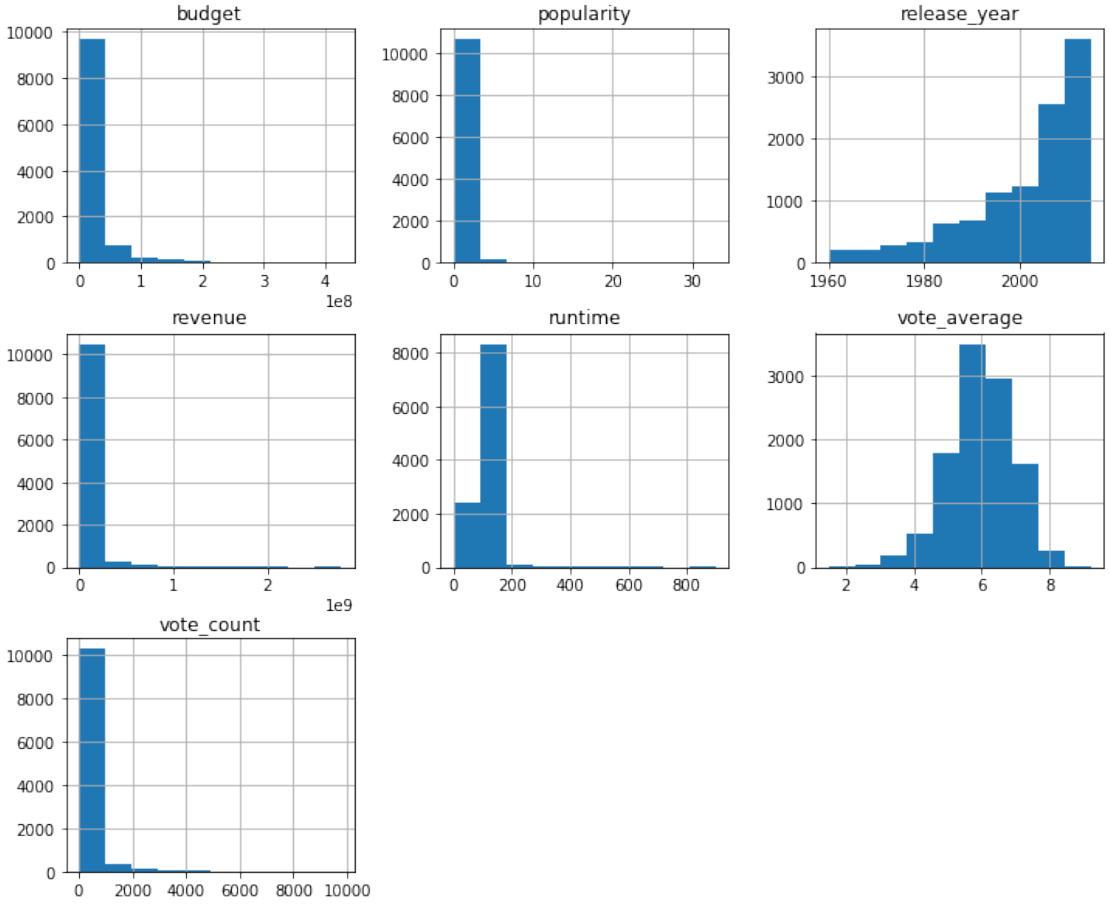
```
In [145]: df.dropna(inplace=True)
```

```
In [146]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10843 entries, 0 to 10865
Data columns (total 8 columns):
popularity      10843 non-null float64
budget          10843 non-null int64
revenue         10843 non-null int64
runtime         10843 non-null int64
genres          10843 non-null object
vote_count      10843 non-null int64
vote_average    10843 non-null float64
release_year    10843 non-null int64
dtypes: float64(2), int64(5), object(1)
memory usage: 762.4+ KB
```

```
In [147]: df.hist(figsize=(12,10))
```

```
Out[147]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f53a684ecc0>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a64d7080>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a60af080>],
                 [<matplotlib.axes._subplots.AxesSubplot object at 0x7f53a5c7f940>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a58a55c0>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a58a5940>],
                 [<matplotlib.axes._subplots.AxesSubplot object at 0x7f539d9c9518>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a4d04400>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x7f53a471e940>]], dtype=obje
```

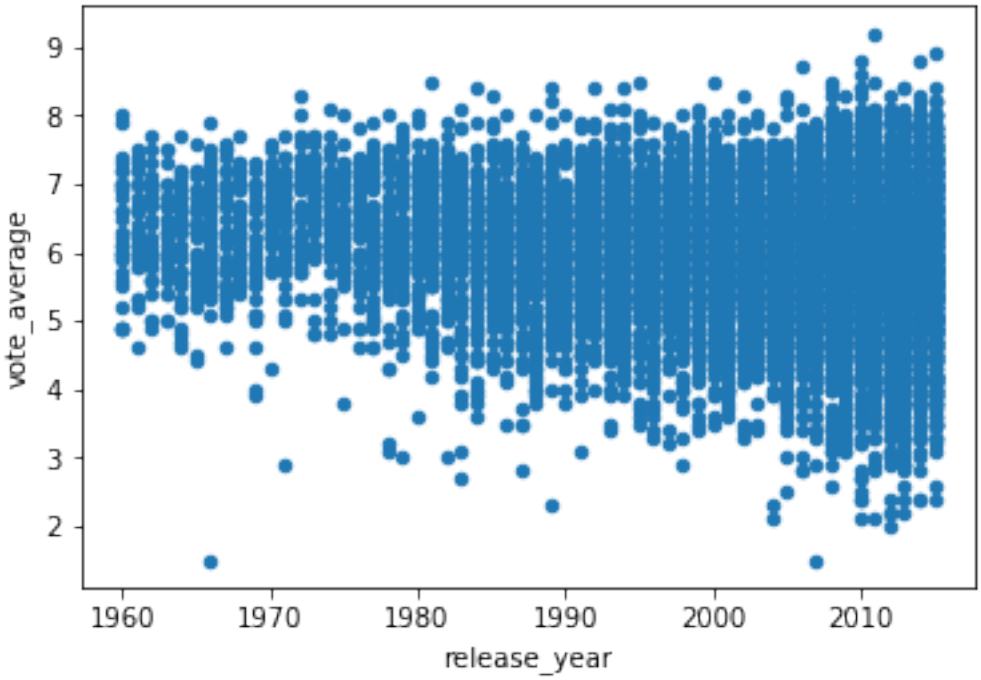


Exploratory Data Analysis

1.1.3 Research Question 1: Is recently released movies come with higher rating point?

```
In [163]: df.plot(x='release_year', y = 'vote_average', kind = 'scatter')
```

```
Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x7f538536c978>
```

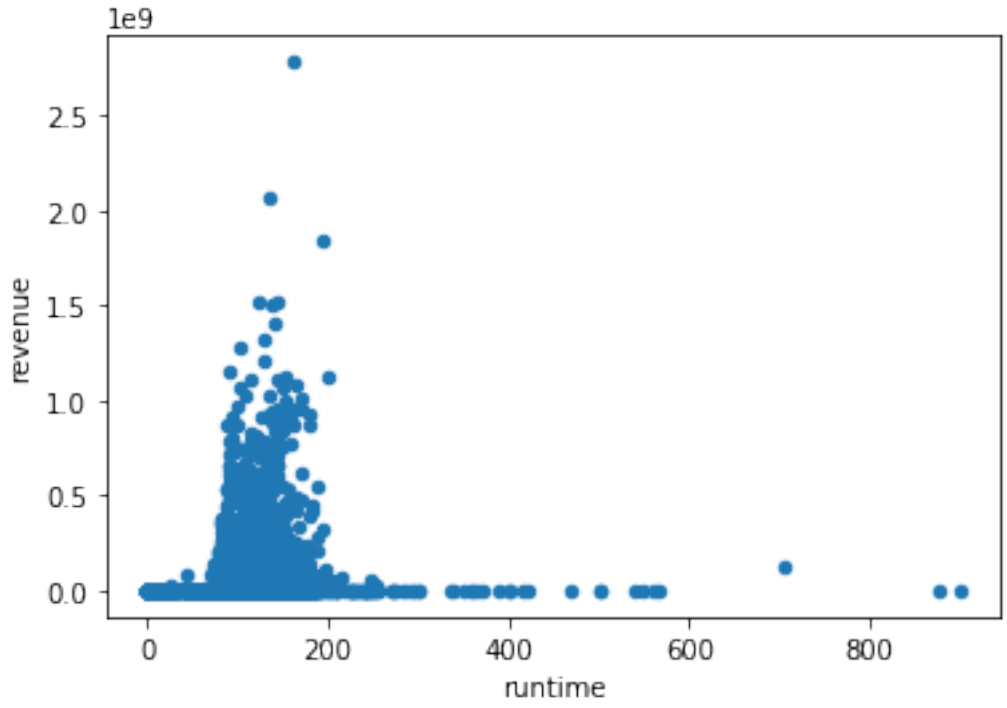


Answer: Not quite. As illustrated in the scatter graph, recently released movies come with both higher and lower range of rating point compared to the old movies. In other words, customer rating are now more diversity than that of the past.

1.1.4 Research Question 2: What is the sweet spot for a movie length? (of which they make the most revenue)

```
In [153]: df.plot(x = 'runtime', y='revenue', kind='scatter')
```

```
Out[153]: <matplotlib.axes._subplots.AxesSubplot at 0x7f53a07d6278>
```



As demostrated by the graph, the most profitable movies were created in between 130 to 200 minutes.

Conclusions

Based on my investigation, a good movie should go between 130 to 200 minutes to be able to make the most profit.

More ever, customer rating are now more diversity than that of the past.

1.2 Submitting your Project

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [150]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[150]: 0
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: